R Packages for Knowledge Space Theory

Cord Hockemeyer[1]
Institute of Psychology, University of Graz, Austria

Second Edition; June 6, 2024

[1]Email: cord.hockemeyer@uni-graz.at

Abstract

R is a statistical software and programming language. It provides a vast selection of statistical and graphical techniques, and it is highly extensible.

Knowledge space theory is a psychological model for structuring domains of knowledge through prerequisite relationships. Originally developed having the adaptive and parsimonious assessment of knowledge in mind, its application has been moving more and more towards computer–based personalised learning.

Over the last 15 years, several R packages for knowledge space theory have been developed by various authors. This report attempts to give an overview on these packages.

Contents

1 Introduction

R (R Core Team, 2022) is an open–source environment for statistical computing and graphics. One of the sources of its success is the possibility to augment its functionality by additional packages. Currently (Oct 2023), alone the Comprehensive R Archive Network (CRAN) contains almost 20,000 such packages — not counting other repositories like GitHub[1] with over 30,000 packages.

Knowledge space theory (KST) was founded by Doignon and Falmagne (1985, 1999, see also Falmagne et al., 2013) as a behavioural model having an efficient adaptive assessment in mind. The core idea is to structure a domain of knowledge based on prerequisite relationships. These prerequisite relationships help to reduce the number of possible knowledge states drastically.

Generally it should be noted that knowledge structures can grow very large. Then R might be not the optimal environment for computations — as an interpreter language it might sometimes lack the necessary speed.

In Section 2, a short introduction to KST is given. Section 3 presents the R packages `kst`, `kstMatrix`, and `kstIO` which provide some general functionality. More specific functionalities are provided by the packages

[1]`https://github.com/topics/r`

`pks` and `DAKS` described in Section 4. Both sections, however, do not aim to replace any documentation of the packages; instead they shall give an idea of what is available. In the final Section 5, some conclusions are drawn and an outlook on possible future developments is given.

A typographic note: within this document, package and function names are written in `typewriter` font to make identification easier.

2 Theoretical background

This section comprises the core ideas of knowledge space theory (KST Doignon and Falmagne, 1985, 1999).

A domain of knowledge is characterised by a set Q of items. The *knowledge state* of a learner is the subset $K \subseteq Q$ of items this learner masters. The family of all knowledge states possible within a population is called *knowledge structure* $\mathcal{K} \subseteq 2^K$. Knowledge structures contain at least the empty state $\emptyset$ and the full knowledge state Q.

The set of possible knowledge states may be restricted by prerequisite or precedence relationships. These are formalised by the *surmise relation*, mathematically a quasi–order. For two items $a, b \in Q$, we write $a \preceq b$ if, from a learner's mastery of b, we can surmise his/her mastery of a. A knowledge structure following the restrictions of a surmise relation is called a *quasi-ordinal knowledge space*, it is a knowledge structure closed under union and intersection, i. e. for any two knowledge states $K, K' \in \mathcal{K}$, their union $K \cup K'$ and their intersection $K \cap K'$ are also knowledge states in $\mathcal{K}$.

Surmise relations have one weakness: they do not allow to model items with different solution paths involving different sets of prerequisites. This is covered by attribution functions and surmise systems. An *attribution function* is a function $\sigma : Q \to 2^{2^Q \setminus \emptyset}$. The $C \in \sigma(q)$ are called *clauses of* q. The different clauses for an item are alternative sets of prerequisites.

A special type of attribution functions are *surmise systems.* A surmise system is an attribution function for which the following conditions hold:

[R] For all $q \in Q$ and $C \in \sigma(q)$, $q \in C$ (extended reflexivity).

[T] For all $q \in Q$ and $q' \in C \in \sigma(q)$, there exists a $C' \in \sigma(q')$ such that $C' \subseteq C$ (extended transitivity).

[I] For all $q \in Q$ and $C, C' \in \sigma(q)$, if $C \subseteq C'$ then $C = C'$ (incomparability).

Surmise mappings correspond to *knowledge spaces*, i. e. knowledge structures closed under union. The clauses $C \in \sigma(q)$, $q \in Q$ are also called *atoms of* q; they are the minimal knowledge states containing q.

Surmise relations can be seen as special cases of surmise systems. For a surmise relation $\preceq$, a surmise system $\sigma_{\preceq}$ can be uniquely defined by $\sigma_{\preceq}(q) = \{q' \in Q \mid q' \preceq q\}$. Vice versa, for a surmise mapping with $|\sigma(q)| = 1$ for all $q \in Q$, we can derive a surmise relation $\preceq_\sigma$ by $q \preceq_\sigma q'$ if $q \in C \in \sigma(q')$.

There is a one–to–one correspondence between surmise relations and quasi–ordinal knowledge spaces and between surmise systems and knowledge spaces over a given set Q of items.

For a knowledge space $\mathcal{K}$, there exists a minimal collection of knowledge states from which the space can be rebuilt by closure under union. This collection is called *Basis* (or base) of $\mathcal{K}$, it is denoted as $\mathcal{B}$. The basis of a knowledge space can easily be determined through the corresponding surmise system: $\mathcal{B} = \bigcup_{q \in Q} \sigma(q)$.

3 General functionalities for KST

The packages described in this section mainly offer general functionalities like conversion between the different representations for knowledge structures.

3.1 The R package `kst`: Knowledge space theory

The `kst` package (Stahl et al., 2022; Stahl, 2008, originally developed by Stahl and Meyer, and now maintained by Hockemeyer) provides functions for working with knowledge structures. These functions are technically realised very close to the theory, i. e. they use an R implementation of sets and relations (Meyer and Hornik, 2022b, 2009, 2022a) for their internal representation of the knowledge structures.

`kst` makes strong use of object oriented classes thus making sure that functions are called with the right type of data. The following classes are defined:

`kbase` Basis

`kfamset` Family of sets

`kstructure` Knowledge structure

kspace Knowledge space

Furthermore, the classes `set` and `relation` from the respective R packages are used. Please note that usually objects of classes `kbase` and `kstructure` are also objects of `kfamset`, and `kspace` objects also have `kstructure` (and `kfamset`).

Most functions names of `kst` start with a `"k"` thus avoiding name conflicts with other functions/packages. The package offers a multitude of functions which are summarised in groups subsequently.

Converting between R representations The functions `as.binaryMatrix()` and `as.famset()` convert knowledge structures between set and matrix representation.

Structural representations `kbase()` computes the basis of a knowledge space. `kstructure()` creates a knowledge structure from a famset of a relation. `kspace()` computes the knowledge space for a famset, i. e. its closure under union. Finally, the method `as.relation()` (for classes `kbase`, `kfamset`, `kstructure`, and `kspace`) determines the surmise relation for a collection of states.

Properties of knowledge structures `katoms()` determines the atoms, i. e. the minimal states of a knowledge structure, `kdomain()` returns the domain of a structure, `knotions()` gives its notions.

The functions `kfringe()` and `kneighbourhood()` determine fringe and neighbourhood of a knowledge state, i. e. the collection of states with a set difference of "1" and the set of items building the difference between a state and its neighbours.

`kstructure_is_wellgraded()` returns whether a knowledge structure is well–graded, functionkstructure_is_kspace() determines whether a structure is a space, i. e. closed under union.

Working with response patterns `kassess()` does a deterministic knowledge assessment for a given response pattern. `kvalidate()` computes several validation measures for a structure and a set of response patterns (as matrix).

Learning paths `lpath()` determines the learning paths in a knowledge structure, `lpath_is_gradation()` tells you whether a certain learning path is a gradation.

Further functions `closure()` computes the closure under union or intersection of a knowledge structure, the `reduction()` method (for classes `kspace`, `kstructure`, and `kfamset`) is its opposite. `ktrace()` determines the *trace* of a knowledge structure, i. e. the restriction to a subset of items.

Furthermore, the `plot()` method has been implemented for the classes `kfamset`, `kbase`, and `kstructure`. It plots a Hasse diagram of the respective family of sets.

3.2 The R package `kstMatrix`: Basic functions in knowledge space theory using matrix representation

The `kstMatrix` package (Hockemeyer and Wong, 2023) has a high overlap in functionality to the `kst` package. However, to achieve better computational performance, it uses a matrix representation which is more native to R than sets and relations.

To avoid name collisions with other packages, all functions offered by `kstMatrix` have a name starting with `km`.

The functions of `kstMatrix` can be summarised in the following groups:

Structure representations The `kmbasis()` function determines the basis of a knowledge structure/space. `kmunionclosure()` is its opposite, i. e. it determines the smallest knowledge space containing a given collection of knowledge states. `kmsurmiserelation()` computes the surmise relation corresponding to the smallest quasi–ordinal knowledge space containing a given collection of knowledge states.

Properties of knowledge structures `kmfringe()` computes the *fringe* of a knowledge state, i. e. the items distinguishing the state from its *neighbours.* It uses the `kmneighbourhood()` function which computes the collection of neighbours for a given state, i. e. all other states with a distance of 1.

`kmiswellgraded()` returns a Boolean value indicating whether a knowledge space is *well–graded.*

`kmnotions()` identifies *notions* in a knowledge structure, i. e. equivalent items.

Graphics The `kmhasse()` function draws a coloured Hasse diagram of a knowledge structure. A respective colour vector can be defined through the `kmcolor()` function.

Simulation and validation `kmsimulate()` simulates a set of response patterns based on a knowledge structure and simulation parameters according to the *BLIM* (Basic Local Independence Model).

`kmvalidate()` returns several validation measures indicating how well a given data set (response patterns) fits to a given knowledge structure. It uses the `kmdist()`, `kmsymmsetdiff()`, and `kmsetdistance()` functions for this purpose.

Data `kstMatrix` offers several data sets, `cad`, `readwrite`, `fractions`, and `xpl`. The first three data sets are from the former research group around Cornelia Dowling at Braunschweig, Germany, the latter one is a small example structure. All these structures can be activated with the `data(<dataset-name>)` command.

3.3 The R package `kstIO`: Knowledge space theory input/Output

The `kstIO` package (Hockemeyer, 2023b) offers functions to read and write knowledge structure files to be used with the `kst` and `kstMatrix` packages, i. e. it works with sets and matrix representations likewise.

`kstIO` supports three different groups of file formats, matrix, KST, and SRBT.

The **matrix** format is simply a binary ASCII matrix where a "1" in row i and column j means that item j is an element of state/response pattern i ("0" otherwise).

There is no space between the rows, and there should be no trailing whitespaces at the start or end of the lines. The last line of the matrix must carry an EndOfLine — in most text editors (except, e. g., `vi`) this means an empty line at the end of the file.

The **KST** format (Hockemeyer and Dowling, 1996; Hockemeyer, 2001) extends the matrix format by two preceding header lines containing the number of items and the number of states/response patterns.

The **SRBT** format (Pötzi and Wesiak, 2001) extends the KST format by yet another preceding header line specifying format and content metadata. This additional header line has the format

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

where `<struct>` specifies the type of data stored in the file and `<comment>` is an arbitrary comment. The following file types are supported by the respective `kstIO` functions:

- basis

- data
- relation
- space
- structure

Basis files are only available in the KST and SRBT formats. Their matrices may also contain "2"s: A "1" means the the state is minimal for the item, and a "2" means that the state contains the item but is not minimal for it.

(Surmise) relation files are available only in the matrix and SRBT formats. They are in a certain sense transposed in comparison to the other files: A "1" in row i and column j means that knowing i can be surmised from knowing j ("0"§ otherwise). Thus, effectively column j describes the minimal state for item j.

4 Specific functionalities for KST

These packages aim at more specific topics within knowledge space theory.

4.1 The R package `pks`: Probabilistic knowledge structures

The `pks` (Heller and Wickelmaier, 2013; Wickelmaier et al., 2022) package provides several functionalities and data sets.

BLIM The BLIM (Basic Local Independence Model) is the generally used probabilistic model in knowledge space theory. It assumes that probabilities for lucky guesses and careless errors are item properties, i. e. independent of the subject's knowledge state.

The `blim()` function fits a BLIM for a knowledge structure and a vector of response pattern frequencies. It can then be printed or plotted with the `print()` and `plot()` methods.

The `jacobian()` computes the Jacobian matrix for a BLIM. The `residuals()` method computes deviance and Pearson residuals for `blim` objects, and the `simulate()` method simulates response patterns according to the given BLIM.

Building knowledge structures `delineate()` computes the knowledge structure delineated by a skill function. The `ita()` function performs an item tree analysis (ITA) on a set of response patterns.

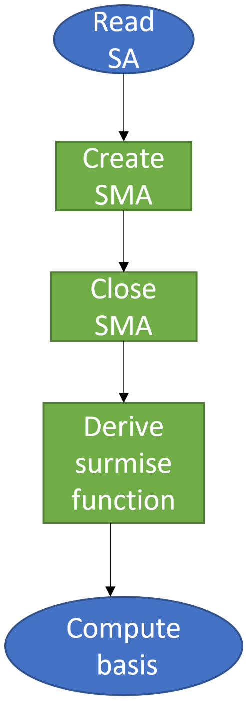

Figure 1: CDSS Workflow

Gradedness The functions `is.forward.graded()` and `is.backward.graded()` check the forward– and backward–gradedness, respectively, of a knowledge structure.

Conversion The `as.pattern()` and `as.binmat()` functions offer several conversion functionalities.

Data sets `pks` offers several data sets, `chess`, `probability`, `density97`, `matter97`, `DoignonFalmagne7`, and `endm`. The first four data sets are empirical ones, the latter two are fictitious ones.

4.2 The R package CDSS: Course–dependent skill structures

`CDSS` (Hockemeyer, 2023a) implements functions deriving skill structures from an assignment of taught and required skills to learning objects in an existing course. There is a rahter straight workflow shown in Fig. 1.

Assuming there exists a pair of tables specifying the skills taught in and required by, respectively, the learning objects, the process to obtain a basis for the skill space consists of the following function calls:

1. `read_skill_assignment_XXX()` reads a skill assignment from tables in XXX format (maybe .csv, .ods, or .xlsx) and checks if it fulfils the CDSS requirements.

   This function makes use of some other functions provided by `CDSS`.

2. `cdss_sa2sma()` creates a skill multi–assignment from the skill assignment read before.

3. `cdss_sma2csma()` closes the skill multi–assignment under completion.

4. `cdss_csma2sf` derives a surmise function from the complete skill multi–assignment.

5. `cdss_sf2basis` simplifies the surmise function to a basis.

The resulting basis of the skill space can then be further processed, e. g. with the functions from the `kstIO` and `kstMatrix` packages.

4.3 The R package `DAKS`: Data analysis and knowledge spaces

`DAKS` (Ünlü and Sargin, 2010; Sargin and Ünlü, 2016) implements functions for analysing and simulating data based on item tree analysis (ITA). Although all functions are public, some of them are not necessarily meant to be called by the user but by other, higher level functions. Subsequently, the `DAKS` functions are summarised:

IITA Inductive Item Tree Analysis: Functions `corr_ita()`, `mini_iita()`, functionorig_iita(), `pop_iita()`, and `iita()`. `ind_gen()` generates a set of competing quasi orders.

Simulation `simu()` does BLIM simulation.

Helper functions `hasse` draws a Hasse diagram for a surmise relation. `pattern()` counts the frequency of patterns in a data set.

Conversion `imp2state()` determines the quasi–ordinal knowledge space for a given surmise relation, `state2imp()` does the opposite.

`print()` method `print()` methods for objects of classes `iita`, `popiita`, `summpopiita`, `ztest`, and `pat`.

`summary()` method Methods for the classes `iita` and `popiita`.

Data set `data(pisa)` provides a partial PISA dataset with responses of 340 German students on a mathematical literacy test with 5 items.

5 Conclusions

The R packages for KST offer already a multitude of functionalities but more will likely come. The author is, e. g. currently developing a package with functions for building course dependent skill structures.

There exists, however, one major problem with using R in KST: real structures can easily grow very large, and R is not always the fastest programming environment. One solution my be to re–implement certain functions in C/C++ and include these into the R packages. This holds especially for the functions computing a knowledge space from a basis or, more generally, closing a collection of sets under union or intersection.

References

Doignon, J.-P. and Falmagne, J.-C. (1985). Spaces for the assessment of knowledge. *International Journal of Man-Machine Studies*, 23:175–196.

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

Falmagne, J.-C., Albert, D., Doble, C., Eppstein, D., and Hu, X., editors (2013). *Knowledge Spaces: Applications in Education*, Heidelberg. Springer.

Heller, J. and Wickelmaier, F. (2013). Minimum discrepancy estimation in probabilistic knowledge structures. *Electronic Notes in Discrete Mathematics*, 42:49–56.

Hockemeyer, C. (2001). KST tools user manual. Unpublished technical report, Institut für Psychologie, Karl-Franzens-Universität Graz, Austria. `https://kst.hockemeyer.at/techreports/KST-Tools_TechRep_FWF01.pdf`.

Hockemeyer, C. (2023a). *CDSS: Course–Dependent Skill Structures*. R package version 0.2-0.

Hockemeyer, C. (2023b). *kstIO: Knowledge Space Theory Input/Output*. R package version 0.4-0.

Hockemeyer, C. and Dowling, C. (1996). KST tools user manual. Unpublished technical report, Institut für Psychologie, Technische Universität Braunschweig, Germany. `https://kst.hockemeyer.at/techreports/KST-Tools_TechRep_BS-96.pdf`.

Hockemeyer, C. and Wong, W. (2023). *kstMatrix: Basic Functions in Knowledge Space Theory Using Matrix Representations*. R package version 0.2-0.

Meyer, D. and Hornik, K. (2009). Generalized and customizable sets in R. *Journal of Statistical Software*, 31(2):1–27.

Meyer, D. and Hornik, K. (2022a). *relations: Data Structures and Algorithms for Relations*. R package version 0.6-12.

Meyer, D. and Hornik, K. (2022b). *sets: Sets, Generalized Sets, Customizable Sets and Intervals*. R package version 1.0-21.

Pötzi, S. and Wesiak, G. (2001). SRbT tools user manual. Unpublished technical report, Institut für Psychologie, Karl–Franzens–Universität Graz, Austria. https://kst.hockemeyer.at/techreports/SRBT-Tools_TechRep_FWF01.pdf.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.r-project.org/.

Sargin, A. and Ünlü, A. (2016). *DAKS: Data Analysis and Knowledge Spaces*. R package version 2.1-3.

Stahl, C. (2008). *Developing a Framework for Competence Assessment*. Unpublished dissertation, Vienna University of Economics and Business.

Stahl, C., Meyer, D., and Hockemeyer, C. (2022). *kst: Knowledge Space Theory*. R package version 0.5-3.

Ünlü, A. and Sargin, A. (2010). DAKS: An R package for data analysis methods in knowledge space theory. *Journal of Statistical Software*, 37:1–31.

Wickelmaier, F., Heller, J., Mollenhauer, J., and Anselmi, P. (2022). *pks: Probabilistic Knowledge Structures*. R package version 0.5-0.