

# Package ‘CDSS’

May 7, 2026

**Type** Package

**Title** Course-Dependent Skill Structures

**Version** 0.3-1

**Date** 2026-04-15

**Depends** R (>= 4.3.0)

**Imports** readODS (>= 2.0.0), openxlsx (>= 4.2.0)

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

**License** GPL-3

**NeedsCompilation** no

**Description** Deriving skill structures from skill assignment data for courses (sets of learning objects).

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Suggests** litedown, DiagrammeRsvg, rsvg, png, kstMatrix

**VignetteBuilder** litedown

**Author** Cord Hockemeyer [aut, cre]

**Repository** CRAN

**Date/Publication** 2026-04-27 05:20:02 UTC

## Contents

CDSS . . . . .	2
cdss_binary_matrix_product . . . . .	3
cdss_circular_requirements . . . . .	4
cdss_close_ar . . . . .	5
cdss_csma2sf . . . . .	5
cdss_lo_csma2sf . . . . .	6
cdss_lo_sa2af . . . . .	6
cdss_lo_sa2ar . . . . .	7
cdss_missing_los . . . . .	7

cdss_nonteaching_los . . . . .	8
cdss_read_skill_assignment_csv . . . . .	8
cdss_read_skill_assignment_ods . . . . .	9
cdss_read_skill_assignment_xlsx . . . . .	10
cdss_reduce_sf . . . . .	11
cdss_sa2ar_skill . . . . .	12
cdss_sa2sma . . . . .	12
cdss_sa_compliance . . . . .	13
cdss_sa_describes_sr . . . . .	13
cdss_sma2csma . . . . .	14
cdss_tables2sa . . . . .	14
cdss_wf_read_skill_assignment . . . . .	15

<b>Index</b>	<b>17</b>
--------------	-----------

---

CDSS

*CDSS: Course dependent skill structures*

---

## Description

The CDSS package provides functions for a complete workflow from skill assignment tables to surmise mappings on the sets of skills and learning objects, respectively.

### Suggested workflow for the general case

1. Read the skill assignment using one of the `read_skill_assignments_xxx()` functions.
2. Check the compliance to the definition for skill assignments using `cdss_sa_compliance()`.
3. Convert the skill assignment into a skill multi-assignment using `cdss_sa2sma()`.
4. Close the skill multi-assignment under completion using `cdss_sma2csma()`.
5. Compute the surmise function on skills using `cdss_csma2sf()`.
6. Continue with functions from the `kstMatrix` package, e.g., to obtain a basis and further on a skill space.

### Suggested workflow for the special case of one LO per skill

1. Read the skill assignment using one of the `read_skill_assignments_xxx()` functions.
2. Check whether the skill assignment allows for the derivation of a surmise relation using `cdss_sa_describes_sr()`.
3. If yes, derive an attribution relation from the skill assignment using `cdss_sa2ar_skill()`.
4. Close the attribution relation to a surmise relation using `cdss_close_ar()`.
5. Continue with functions from the `kstMatrix` package, e.g., to obtain a basis and further on a skill space.

## Data files

The installation of this package includes several data files as examples in the `extdata` sub directory (see the Examples below for how to access the files there). There are four data sets, `KST`, `KST-Intro`, `SkillAssignment`, and `ErroneousSkillAssignment`. The `SkillAssignment` data set is available in three formats, `ODS`, `XLSX`, and `CSV` (in `CSV` format, there are two files each, `SkillAssignment-R.csv` and `SkillAssignment-T.csv`, for required and taught skills, respectively). The other three data sets are available in `ODS` format only.

`SkillAssignment` and `ErroneousSkillAssignment` are small example data sets where the latter fails for `cdss_sa_compliance()`. `KST` contains a skill assignment for the course on knowledge space theory under <https://moodle.qhelp.eu/>. `KST-Intro` contains the reduction of `KST` to the first chapter of that course.

## References

Hockemeyer, C. (2022). Building Course-Dependent Skill Structures - Applying Competence based Knowledge Space Theory to Itself. Manuscript in preparation.

## Acknowledgements

The creation of this R package was financially supported by the Erasmus+ Programme of the European Commission through the QHELP project (<https://qhelp.eu/>).

## Examples

```
library(readODS)
fpath <- system.file("extdata", "SkillAssignment.ods", package="CDSS")
sa <- cdss_read_skill_assignment_ods(fpath)
sa
sma <- cdss_sa2sma(sa)
sma
csma <- cdss_sma2csma(sma)
csma
sf <- cdss_csma2sf(csma)
sf
```

---

cdss\_binary\_matrix\_product

*Compute a binary matrix product*

---

## Description

`cdss_binary_matrix_product` expects two binary matrices and computes there Boolean product.

## Usage

```
cdss_binary_matrix_product(m, n)
```

**Arguments**

m	Binary matrix
n	Binary matrix

**Value**

Boolean matrix product of m and n

**See Also**

Other Utility functions: [cdss\\_close\\_ar\(\)](#), [cdss\\_reduce\\_sf\(\)](#)

---

cdss\_circular\_requirements

*Vector of learning objects requiring and teaching the same skill*

---

**Description**

cdss\_circular\_requirements expects skill assignment and returns a vector of learning objects which require a skill that they teach.

**Usage**

```
cdss_circular_requirements(sa)
```

**Arguments**

sa	Skill assignment
----	------------------

**Value**

Vector of learning objects

**See Also**

Other Functions testing validity of skill assignments: [cdss\\_missing\\_los\(\)](#), [cdss\\_nonteaching\\_los\(\)](#), [cdss\\_sa\\_compliance\(\)](#)

---

cdss_close_ar	<i>Close an attribution relation to get a surmise relation.</i>
---------------	---

---

**Description**

cdss\_close\_ar expects a quadratic binary matrix and closes it under reflexivity and transitivity.

**Usage**

```
cdss_close_ar(ar)
```

**Arguments**

ar Attribution relation matrix

**Value**

surmise relation or NULL

**See Also**

Other Utility functions: [cdss\\_binary\\_matrix\\_product\(\)](#), [cdss\\_reduce\\_sf\(\)](#)

---

cdss_csma2sf	<i>Derive a surmise function from a complete skill multi-assignment</i>
--------------	---

---

**Description**

cdss\_csma2sf expects a complete skill multi-assignment object and returns the corresponding surmise function on the set of skills.

**Usage**

```
cdss_csma2sf(csma)
```

**Arguments**

csma Skill multi-assignment to be completed

**Value**

Object of class cdss\_csma.

---

cdss_lo_csma2sf	<i>Derive a surmise function between learning objects from a complete skill multi-assignment</i>
-----------------	--

---

**Description**

cdss\_lo\_csma2sf expects a complete skill multi-assignment and derives a surmise function on the set of learning objects.

**Usage**

```
cdss_lo_csma2sf(csma)
```

**Arguments**

csma                    Complete skill multi-assignment object

**Value**

Object of class cdss\_sf (attribution function).

**See Also**

Other functions building skill (multi) assignment matrices: [cdss\\_lo\\_sa2af\(\)](#), [cdss\\_sa2sma\(\)](#), [cdss\\_tables2sa\(\)](#)

---

cdss_lo_sa2af	<i>Determine Attribution function for LOs from a skill assignment</i>
---------------	---

---

**Description**

cdss\_lo\_sa2af expects a skill assignment and derives an attribution function on the set of learning objects.

**Usage**

```
cdss_lo_sa2af(sa)
```

**Arguments**

sa                      Skill assignment object

**Value**

Object of class cdss\_af (attribution function).

**See Also**

Other functions building skill (multi) assignment matrices: [cdss\\_lo\\_csma2sf\(\)](#), [cdss\\_sa2sma\(\)](#), [cdss\\_tables2sa\(\)](#)

---

cdss_lo_sa2ar	<i>Create an attribution relation on learning objects from a skill assignment.</i>
---------------	--

---

**Description**

cdss\_lo\_sa2ar expects a skill assignment and derives an attribution relation on learning objects if the skill assignment fulfills the necessary conditions, i.e. if there is only one teaching LO per skill.

**Usage**

```
cdss_lo_sa2ar(sa)
```

**Arguments**

sa	Skill assignment object
----	-------------------------

**Value**

attribution relation or NULL

**See Also**

Other functions deriving skill structures from skill assignments: [cdss\\_sa2ar\\_skill\(\)](#), [cdss\\_sa\\_describes\\_sr\(\)](#)

---

cdss_missing_los	<i>Vector of skills without teaching learning objects.</i>
------------------	--

---

**Description**

cdss\_missing\_los expects a skill assignment and returns a vector of skills which are not taught by any learning object.

**Usage**

```
cdss_missing_los(sa)
```

**Arguments**

sa	Skill assignment
----	------------------

**Value**

Vector of skills

**See Also**

Other Functions testing validity of skill assignments: [cdss\\_circular\\_requirements\(\)](#), [cdss\\_nonteaching\\_los\(\)](#), [cdss\\_sa\\_compliance\(\)](#)

---

`cdss_nonteaching_los` *Vector of learning objects not teaching any skills.*

---

**Description**

`cdss_nonteaching_los` expects a skill assignment and returns a vector of learning objects which do not teach any skill.

**Usage**

```
cdss_nonteaching_los(sa)
```

**Arguments**

`sa` Skill assignment

**Value**

Vector of learning objects

**See Also**

Other Functions testing validity of skill assignments: [cdss\\_circular\\_requirements\(\)](#), [cdss\\_missing\\_los\(\)](#), [cdss\\_sa\\_compliance\(\)](#)

---

`cdss_read_skill_assignment_csv`  
*Read an assignment of taught and required skills for a set of learning objects from CSV-files.*

---

**Description**

`cdss_read_skill_assignment` expects two CSV-files with two columns each. The first column contains the IDs of learning objects and the second row the IDs of single skills required or taught, respectively, by this learning object. It returns a list of two binary matrices, "taught" and "required". Each matrix has one row per learning object and one column per skill. The cells contain a "1" if the skill is taught or required, respectively, by the learning object and a "0" otherwise,

**Usage**

```

cdss_read_skill_assignment_csv(
  taught,
  required,
  header = TRUE,
  sep = ",",
  dec = ".",
  warnonly = FALSE,
  verbose = TRUE
)

```

**Arguments**

taught	CSV-file with assignments of taught competencies to learning objects
required	CSV-file with assignments of required competencies to learning objects
header	Boolean specifying whether the CSV-files contain a header line (default = TRUE)
sep	Column separator (default ",")
dec	Decimal point character (default ".")
warnonly	Are non-compliant SAs allowed? (default = FALSE)
verbose	Verbosity of compliance test (default = TRUE)

**Value**

List of two binary matrices, "taught" and "required".

**See Also**

Other functions reading skill assignments: [cdss\\_read\\_skill\\_assignment\\_ods\(\)](#), [cdss\\_read\\_skill\\_assignment\\_xlsx\(\)](#), [cdss\\_wf\\_read\\_skill\\_assignment\(\)](#)

---

cdss\_read\_skill\_assignment\_ods

*Read an assignment of taught and required skills for a set of learning objects from an ODS-file.*

---

**Description**

cdss\_read\_skill\_assignment\_ods expects an ODS-file with two sheets assigning taught and required, respectively, skills to learning objects with two columns each. The first column contains the IDs of learning objects and the second row the IDs of single skills required or taught, respectively, by this learning object. It returns a list of two binary matrices, "taught" and "required". Each matrix has one row per learning object and one column per skill. The cells contain a "1" if the skill is taught or required, respectively, by the learning object and a "0" otherwise,

**Usage**

```
cdss_read_skill_assignment_ods(
  filename,
  taughtname = "Taught",
  requiredname = "Required",
  warnonly = FALSE,
  verbose = TRUE
)
```

**Arguments**

filename	Name of the ODS-file
taughtname	Name of the sheet with required assignment (default = "Taught")
requiredname	Name of the sheet with required assignment (default = "Required")
warnonly	Are non-compliant SAs allowed? (default = FALSE)
verbose	Verbosity of compliance test (default = TRUE)

**Value**

List of two binary matrices, "taught" and "required".

**See Also**

Other functions reading skill assignments: [cdss\\_read\\_skill\\_assignment\\_csv\(\)](#), [cdss\\_read\\_skill\\_assignment\\_xlsx\(\)](#), [cdss\\_wf\\_read\\_skill\\_assignment\(\)](#)

---

cdss\_read\_skill\_assignment\_xlsx

*Read an assignment of taught and required skills for a set of learning objects from an XLSX-file.*

---

**Description**

cdss\_read\_skill\_assignment\_xlsx expects an XLSX-file with two sheets assigning taught and required, respectively, skills to learning objects with two columns each. The first column contains the IDs of learning objects and the second row the IDs of single skills required or taught, respectively, by this learning object. It returns a list of two binary matrices, "taught" and "required". Each matrix has one row per learning object and one column per skill. The cells contain a "1" if the skill is taught or required, respectively, by the learning object and a "0" otherwise,

**Usage**

```

cdss_read_skill_assignment_xlsx(
  filename,
  taughtname = "Taught",
  requiredname = "Required",
  warnonly = FALSE,
  verbose = TRUE
)

```

**Arguments**

filename	Name of the XLSX-file
taughtname	Name of the sheet with required assignment (default = "Taught")
requiredname	Name of the sheet with required assignment (default = "Required")
warnonly	Are non-compliant SAs allowed? (default = FALSE)
verbose	Verbosity of compliance test (default = TRUE)

**Value**

List of two binary matrices, "taught" and "required".

**See Also**

Other functions reading skill assignments: [cdss\\_read\\_skill\\_assignment\\_csv\(\)](#), [cdss\\_read\\_skill\\_assignment\\_ods\(\)](#), [cdss\\_wf\\_read\\_skill\\_assignment\(\)](#)

---

cdss_reduce_sf	<i>Reduce a surmise function with respect to item equivalence</i>
----------------	---

---

**Description**

cdss\_reduce\_sf takes a surmise function and returns its reduction to non-equivalent items.

**Usage**

```
cdss_reduce_sf(sf)
```

**Arguments**

sf	Surmise function
----	------------------

**Value**

Surmise function reduced by equivalences

**See Also**

Other Utility functions: [cdss\\_binary\\_matrix\\_product\(\)](#), [cdss\\_close\\_ar\(\)](#)

---

cdss_sa2ar_skill	<i>Create an attribution relation on skills from a skill assignment.</i>
------------------	--

---

**Description**

cdss\_sa2ar\_skill expects a skill assignment and derives an attribution relation on skills if the skill assignment fulfills the necessary conditions, i.e. if there is only one teaching LO per skill.

**Usage**

```
cdss_sa2ar_skill(sa)
```

**Arguments**

sa	Skill assignment object
----	-------------------------

**Value**

attribution relation or NULL

**See Also**

Other functions deriving skill structures from skill assignments: [cdss\\_lo\\_sa2ar\(\)](#), [cdss\\_sa\\_describes\\_sr\(\)](#)

---

cdss_sa2sma	<i>Convert skill assignment matrices to skill multi-assignment</i>
-------------	--

---

**Description**

cdss\_sa2sma expects a list of two matrices (taught and required) of a skill assignment. It returns a skill multi-assignment object.

**Usage**

```
cdss_sa2sma(sa)
```

**Arguments**

sa	Skill assignment object
----	-------------------------

**Value**

Object of class cdss\_sma.

**See Also**

Other functions building skill (multi) assignment matrices: [cdss\\_lo\\_csma2sf\(\)](#), [cdss\\_lo\\_sa2af\(\)](#), [cdss\\_tables2sa\(\)](#)

---

cdss\_sa\_compliance     *Check whether a skill assignment is compliant to the CDCS conditions.*

---

**Description**

cdss\_sa\_compliance expects a skill assignment and checks whether it is compliant to the conditions for CDCS.

**Usage**

```
cdss_sa_compliance(sa, warnings = FALSE)
```

**Arguments**

sa	Skill assignment
warnings	Toggles whether warnings should be printed

**Value**

Boolean

**See Also**

Other Functions testing validity of skill assignments: [cdss\\_circular\\_requirements\(\)](#), [cdss\\_missing\\_los\(\)](#), [cdss\\_nonteaching\\_los\(\)](#)

---

cdss\_sa\_describes\_sr     *Check whether a surmise relation can be derived from a given skill assignment.*

---

**Description**

cdss\_sa\_describes\_sr expects a list of two matrices (taught and required) of a skill assignment. It returns TRUE if the skill assignment describes a surmise relation (i.e. there is only one teaching LO per skill) and FALSE.

**Usage**

```
cdss_sa_describes_sr(sa, verbose = FALSE)
```

**Arguments**

sa	Skill assignment object
verbose	Flag, default is FALSE

**Value**

Logical value

**See Also**

Other functions deriving skill structures from skill assignments: [cdss\\_lo\\_sa2ar\(\)](#), [cdss\\_sa2ar\\_skill\(\)](#)

---

cdss_sma2csma	<i>Complete a skill multi-assignment</i>
---------------	--

---

**Description**

cdss\_sma2csma expects a skill multi-assignment object and returns the corresponding complete skill multi-assignment. If this would involve cycles, the function stops by default - except if allowcycles is set to TRUE. In that case, the result may be ill-defined!

**Usage**

```
cdss_sma2csma(sma, allowcycles = FALSE)
```

**Arguments**

sma	Skill multi-assignment to be completed
allowcycles	Whether prerequisite cycles should be allowed (default = FALSE)

**Value**

Object of class cdss\_csma.

---

cdss_tables2sa	<i>Build matrices of taught and required, respectively, skills for learning objects from respective tables.</i>
----------------	---

---

**Description**

cdss\_tables2sa expects two data frames with two columns each. The first column contains the IDs of learning objects and the second row the IDs of single skills required or taught, respectively, by this learning object. It returns a list of two binary matrices, "taught" and "required". Each matrix has one row per learning object and one column per skill. The cells contain a "1" if the skill is taught or required, respectively, by the learning object and a "0" otherwise.

**Usage**

```
cdss_tables2sa(taught, required)
```

**Arguments**

taught	Data table containing the assignment of taught skills to learning objects
required	Data table containing the assignment of required skills to learning objects

**Value**

List of two binary matrices, "taught" and "required".

**See Also**

Other functions building skill (multi) assignment matrices: [cdss\\_lo\\_csma2sf\(\)](#), [cdss\\_lo\\_sa2af\(\)](#), [cdss\\_sa2sma\(\)](#)

---

cdss\_wf\_read\_skill\_assignment

*Read an assignment of taught and required skills for a set of learning objects from file and do the whole workflow up to a surmise function on skills*

---

**Description**

cdss\_wf\_read\_skill\_assignment expects an ODS or XLSX file with two sheets assigning taught and required, respectively, skills to learning objects with two columns each. Alternatively, two CSV files can be specified. In the sheets/CSV files, the first column contains the IDs of learning objects and the second row the IDs of single skills required or taught, respectively, by this learning object. It returns a list of two binary matrices, "taught" and "required". Each matrix has one row per learning object and one column per skill. The cells contain a "1" if the skill is taught or required, respectively, by the learning object and a "0" otherwise,

**Usage**

```
cdss_wf_read_skill_assignment(  
  filename,  
  filename2 = NULL,  
  filetype = "auto",  
  taughtname = "Taught",  
  requiredname = "Required",  
  header = TRUE,  
  sep = ",",  
  dec = ".",  
  warnonly = FALSE,  
  verbose = TRUE  
)
```

**Arguments**

filename	Name of the file (in case of CSV files the one with TAUGHT assignments)
filename2	Name of the CSV file with REQUIRED assignments (if applicable)
filetype	Type of the file, allowed values are "auto", "ODS", "XLSX", and "CSV"
taughtname	Name of the sheet with required assignment (default = "Taught")
requiredname	Name of the sheet with required assignment (default = "Required")
header	Boolean specifying whether the CSV-files contain a header line (default = TRUE)
sep	Column separator for CSV files (default ",")
dec	Decimal point character for CSV files (default ".")
warnonly	Are non-compliant SAs allowed? (default = FALSE)
verbose	Verbosity of compliance test (default = TRUE)

**Value**

List of four elements: sfs (surmise function between skills), sfl (surmise function between learning objects) srs (surmise relation between skills, if available; NULL otherwise) srl (surmise relation between learning objects, if available; NULL otherwise)

**See Also**

Other functions reading skill assignments: [cdss\\_read\\_skill\\_assignment\\_csv\(\)](#), [cdss\\_read\\_skill\\_assignment\\_ods\(\)](#), [cdss\\_read\\_skill\\_assignment\\_xlsx\(\)](#)

# Index

- \* **Functions testing validity of skill assignments**
    - cdss\_circular\_requirements, 4
    - cdss\_missing\_los, 7
    - cdss\_nonteaching\_los, 8
    - cdss\_sa\_compliance, 13
  - \* **Utility functions**
    - cdss\_binary\_matrix\_product, 3
    - cdss\_close\_ar, 5
    - cdss\_reduce\_sf, 11
  - \* **functions building skill (multi) assignment matrices**
    - cdss\_lo\_csma2sf, 6
    - cdss\_lo\_sa2af, 6
    - cdss\_sa2sma, 12
    - cdss\_tables2sa, 14
  - \* **functions deriving skill structures from skill assignments**
    - cdss\_lo\_sa2ar, 7
    - cdss\_sa2ar\_skill, 12
    - cdss\_sa\_describes\_sr, 13
  - \* **functions reading skill assignments**
    - cdss\_read\_skill\_assignment\_csv, 8
    - cdss\_read\_skill\_assignment\_ods, 9
    - cdss\_read\_skill\_assignment\_xlsx, 10
    - cdss\_wf\_read\_skill\_assignment, 15
  - cdss\_nonteaching\_los, 4, 8, 8, 13
  - cdss\_read\_skill\_assignment\_csv, 8, 10, 11, 16
  - cdss\_read\_skill\_assignment\_ods, 9, 9, 11, 16
  - cdss\_read\_skill\_assignment\_xlsx, 9, 10, 10, 16
  - cdss\_reduce\_sf, 4, 5, 11
  - cdss\_sa2ar\_skill, 7, 12, 14
  - cdss\_sa2ar\_skill(), 2
  - cdss\_sa2sma, 6, 7, 12, 15
  - cdss\_sa2sma(), 2
  - cdss\_sa\_compliance, 4, 8, 13
  - cdss\_sa\_compliance(), 2
  - cdss\_sa\_describes\_sr, 7, 12, 13
  - cdss\_sa\_describes\_sr(), 2
  - cdss\_sma2csma, 14
  - cdss\_sma2csma(), 2
  - cdss\_tables2sa, 6, 7, 12, 14
  - cdss\_wf\_read\_skill\_assignment, 9–11, 15
- CDSS, 2
- CDSS-package (CDSS), 2
- cdss\_binary\_matrix\_product, 3, 5, 11
- cdss\_circular\_requirements, 4, 8, 13
- cdss\_close\_ar, 4, 5, 11
- cdss\_close\_ar(), 2
- cdss\_csma2sf, 5
- cdss\_csma2sf(), 2
- cdss\_lo\_csma2sf, 6, 7, 12, 15
- cdss\_lo\_sa2af, 6, 6, 12, 15
- cdss\_lo\_sa2ar, 7, 12, 14
- cdss\_missing\_los, 4, 7, 8, 13