UNIVERSITY OF GRAZ
Department of Psychology

kstpy

Cord Hockemeyer

Mar 18, 2025

Contents

A Python package for Knowledge Space Theory

1 General package informaation

1.1 Remarks

- Please note that is an early stage package.
- Especially the functions in the `kstpy.basics` module may be rather inefficiednt.

1.2 Installation from PyPI

```
$ pip install kstpy
```

1.3 Usage

```
from kstpy.data import xpl_base
from kstpy.basics import constr

print(constr(xpl_base))
```

For further information, please have a look at the API reference.

1.4 License

`kstpy` was created by Cord Hockemeyer. It is licensed under the terms of the GNU General Public License v3.0 license.

1.5 Credits

- The `graphicx` module is a copy of Simon Hegele's `hasseNetworkx` module from GitHub. I opted for this way because I did not find any comparable Python package.
- `kstpy` was created with `cookiecutter` and the `py-pkgs-cookiecutter` template.

Example usage

To use `kstpy` in a project:

```
import kstpy

print(kstpy.__version__)
```

```
0.1.1
```

Changelog

v0.2.0 (18/03/2025)

- Adding various functions (equivalence, srdomain, reduceSR, sr2basis, hasse, sr_hasse, simulate, fringe & neighbourhood, gradationsd, learningpaths)

v0.1.0 (13/03/2025)

- Adding I/O functions
- Adding functions basis and surmiserelation

v0.0.6 (12/03/2025)

- Improving metadata

v0.0.5 (11/03/2025)

- Restructuring: replacing sub packages by modules
- Adding more validation functions (di, da)

v0.0.4 (11/03/2025)

- Validation functions distvec & difreq

v0.0.3 (10/03/2025)

- Documentation

v0.0.2 (10/03/2025)

- Sub packages basics, data, and helpers
- Functions/modules constr, itemname, and domain

v0.0.1 (08/03/2025)

- First release of `kstpy`!

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

Please send your suggestions to cord.hockemeyer@uni-graz.at

Types of Contributions

Report Bugs

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Suggest improvements/extensions

The kstpy package is still in early development. Ideas for improvements and extensions are heavily welcome.

Code of Conduct

Please note that the `kstpy` project is released with a Code of Conduct. By contributing to this project you agree to abide by its terms.

Code of Conduct

Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

Attribution

This Code of Conduct is adapted from the Contributor Covenant homepage, version 1.4.

API Reference

This page contains auto-generated API reference documentation[1].

kstpy

Submodules

kstpy.basics

Created on Sun Feb 23 2025

@author: Cord Hockemeyer

[1] Created with sphinx-autoapi

Functions

constr(structure)	Compute the smallest knowledge space containing a famly of kstFrozensets
basis(structure)	Determine the basis of a knolwdge space/structure
surmiserelation(structure)	Compute the surmise relation for a knowledge structure
sr2basis(sr)	Compute the basis corresponding to a surmise relation
neighbourhood(state, structure[, maxdist])	Determine the neighbourhood of a state"
fringe(state, structure[, maxdist])	Determine the inner, outer, and total fringe of a state"
equivalence(structure)	Determine equi9valence classes
gradations(s1, s2, structure)	Determine all gradations from s1 to s2 in structure
learningpaths(structure)	Return all learning paths in a knwoledge structure

Module Contents

kstpy.basics.**constr**(*structure*)

Compute the smallest knowledge space containing a famly of kstFrozensets

Parameters

structure (*set*) – Family of kstFrozensets

Return type

Knowledge space

Examples

```
>>> from kstpy.data import xpl_basis
>>> constr(xpl_basis)
```

kstpy.basics.**basis**(*structure*)

Determine the basis of a knolwdge space/structure

Parameters

structure (*set*) – Family of kstFrozensets

Return type

Basis

Examples

```
>>> from kstpy.data import xpl_basis
>>> s = constr(xpl_basis)
>>> basis(s)
```

kstpy.basics.**surmiserelation**(*structure*)

Compute the surmise relation for a knowledge structure

Parameters
structure (*set*) – Family of kstFrozensets

Return type
Corresponding surmise relation

Examples

```
>>> from kstpy.data import xpl_basis
>>> s = constr(xpl_basis)
>>> surmiserelation(s)
```

kstpy.basics.**sr2basis**(*sr*)

Compute the basis corresponding to a surmise relation

Parameters
sr (*set (of 2-tuples)*) –

Return type
Corresponding basis

kstpy.basics.**neighbourhood**(*state*, *structure*, *maxdist=1*)

Determine the neighbourhood of a state”

Parameters

- **state** (kstFrozenset) –
- **structure** (*set (of kstFrozensets)*) –
- **maxdist** (*int (radius of the neighbourhood; default = 1)*) –

Return type
Set containing the neighbourhood

kstpy.basics.**fringe**(*state*, *structure*, *maxdist=1*)

Determine the inner, outer, and total fringe of a state”

Parameters

- **state** (kstFrozenset) –
- **structure** (*set (of kstFrozensets)*) –
- **maxdist** (*int (radius of the fringe; default = 1)*) –

Returns
Dictionary with three lists

Return type
fringe, inner (fringe), and Outer (fringe)

kstpy.basics.**equivalence**(*structure*)

Determine equi9valence classes

Parameters
structure (*set or list of kstFrozensets*) –

Return type
Equivalence classes (set of kstFrozensets)

kstpy.basics.**gradations**(*s1*, *s2*, *structure*)

Determine all gradations from s1 to s2 in structure

Parameters

- **s1** (kstFrozenset *(starting state)*) –
- **s2** (kstFrozenset *(goal state)*) –
- **structure** (*set of kstFrozensets (knowledge structure)*) –

Return type
set of lists containing all gradations from s1 to s2 within structure

kstpy.basics.**learningpaths**(*structure*)

Return all learning paths in a knwoledge structure

Parameters
structure (*set of kstFrozensets*) –

Return type
set of lists containing all learning paths in the structure

kstpy.data

Created on Sun Feb 23 2025

@author: Cord Hockemeyer

Attributes

xpl_basis

Module Contents

kstpy.data.**xpl_basis**

kstpy.graphics

Functions

_exists_path(Graph, u, v[, start])	If there exits a path from u to v in G with length > 0: Return True
_transitivity_elimination(Graph)	Removes edges that are implied by transitivity of the partial order
_layer(positions, i)	Returns the positions of nodes at layer i in a dictionary
_y_positioning(Graph, positions[, n])	
_y_positioning_by_function(Graph, positions, ...)	
_number_of_layers(positions)	Returns the number of layers in the hasse-diagramm
_max_layer_size(positions)	Returns the maximum numbber of nodes in any layer of the hasse-diagram
_shift_x_positions(positions)	Shifts the layers alternately by half a unit to the left or right along the x-axis.
_x_positioning(Graph, positions[, shift_x])	
_layout(Graph[, layer_function, shift_x])	Returns a dictionary with positions for the nodes of the graph
hasse(structure[, title])	Draw a Hasse diagram for a knowledge structure
sr_hasse(sr[, title])	Draw a Hasse diagram for a surmise relation

Module Contents

kstpy.graphics.**_exists_path**(*Graph*, *u*, *v*, *start=True*)

If there exits a path from u to v in G with length > 0: Return True Else : Return False Recursive method (Checks if there exists a path from any successor u' of u to v)

Parameters:

Graph (networkx.DiGraph) u (string), identifier of first node in Graph v (string), identifier of second node in Graph start (boolean), True if u is the recursion start

False else

Returns:

(boolean) True if there exits a path from u to v in G

False else

Additional remark:

If the execution of this function leads to an stack overflow this can indicate a faulty method for the evaluation of the relationships that might cause circles in the Graph.

kstpy.graphics.**_transitivity_elimination**(*Graph*)

Removes edges that are implied by transitivity of the partial order

Parameters:

Graph (networkx.DiGraph)

Returns:

Additional remark:

If the execution of this function leads to an stack overflow this can indicate a faulty method for the evaluation of the relationships that might cause circles in the Graph.

kstpy.graphics.**_layer**(*positions*, *i*)

Returns the positions of nodes at layer i in a dictionary

Parameters:

posititions (dictionary) i (int)

Returns:

(dictionary) the positions of nodes at layer i in a dictionary

kstpy.graphics.**_y_positioning**(*Graph*, *positions*, *n=0*)

kstpy.graphics.**_y_positioning_by_function**(*Graph*, *positions*, *layer_function*)

kstpy.graphics.**_number_of_layers**(*positions*)

Returns the number of layers in the hasse-diagramm (i.e. the highest y-position value in positions)

Parameters:

positions (dictionary)

Returns:

(int)

kstpy.graphics.**_max_layer_size**(*positions*)

Returns the maximum numbber of nodes in any layer of the hasse-diagram

Parameters:

positions (dictionary)

Returns:

(int)

kstpy.graphics.**_shift_x_positions**(*positions*)

Shifts the layers alternately by half a unit to the left or right along the x-axis.

Parametrs:

positions (dictionary)

Returns

(dictionary)

kstpy.graphics.**_x_positioning**(*Graph*, *positions*, *shift_x=False*)

kstpy.graphics.**_layout**(*Graph*, *layer_function=None*, *shift_x=False*)

Returns a dictionary with positions for the nodes of the graph

kstpy.graphics.**hasse**(*structure*, *title=''*)

Draw a Hasse diagram for a knowledge structure

Parameters

- **structure** (*list or set of sets (or derivatives)*) –
- **title** (*str (optional)*) –
- **Result** –
- **------** –
- **diagram** (*Plot of Hasse*) –

Example

```
>>> hasse(kstpy.data.xpl_basis, title = "Small example basis")
```

kstpy.graphics.**sr_hasse**(*sr*, *title=''*)

Draw a Hasse diagram for a surmise relation

Parameters

- **sr** (*set of 2-tuples (surmise relation)*) –
- **title** (*str (optional)*) –
- **Result** –
- **------** –
- **diagram** (*Plot of Hasse*) –

Example

```
>>> sr_hasse(kstpy.basics.surmiserelation(kstpy.data.xpl_basis),␣
↪title = "Small example")
```

kstpy.helpers

Created on Sun Mar 9 13:03:53 2025

@author: hockemey

Classes

kstFrozenset	kstFrozenset is a derivative of the frozenset class.

Functions

itemname(num)	Return an Itemname based on the item number
domain(structure)	Determine the domain of a set/list of frozensets
srdomain(sr)	Determine the domain of a surmise relation
reduceSR(sr)	Remove transitivities from a (surmise) relation
powerset(domain)	
vector2kstFrozenset(v, d)	
kstFrozenset2vector(s, d)	

Module Contents

class kstpy.helpers.**kstFrozenset**

Bases: frozenset

kstFrozenset is a derivative of the frozenset class.

The main reason for its existance is the print function which does not show the class anymore. Set operands (|, &, ^, and -) have also been re-defined to produce kstFrozensets.

__repr__()
Return repr(self).

__str__()
Return str(self).

__or__(*value*)
Return self|value.

__and__(*value*)
Return self&value.

__sub__(*value*)
Return self-value.

__xor__(*value*)
Return self^value.

kstpy.helpers.**itemname**(*num*)
Return an Itemname based on the item number

kstpy.helpers.**domain**(*structure*)
Determine the domain of a set/list of frozensets

kstpy.helpers.**srdomain**(*sr*)
Determine the domain of a surmise relation

kstpy.helpers.**reduceSR**(*sr*)
Remove transitivities from a (surmise) relation

Parameters
sr (*set (of 2-tuples)*) –

Return type
Reduced relation

kstpy.helpers.**powerset**(*domain*)

kstpy.helpers.**vector2kstFrozenset**(*v*, *d*)

kstpy.helpers.**kstFrozenset2vector**(*s*, *d*)

kstpy.io

Created on Thu Mar 13 2025

@author: Cord Hockemeyer

Functions

readpatterns(filename)	Read a set of response patterns from a file
readstructure(filename)	Read a knowledge structure from a file
writekst(filename, x)	Write a knowledge structure or a data set to a file

Module Contents

kstpy.io.**readpatterns**(*filename*)

Read a set of response patterns from a file

Parameters

filename (*str*) –

Return type

List of patterns (kstFrozensets)

kstpy.io.**readstructure**(*filename*)

Read a knowledge structure from a file

Parameters

filename (*str*) –

Return type

Set of states (kstFrozensets)

kstpy.io.**writekst**(*filename*, *x*)

Write a knowledge structure or a data set to a file

Parameters

- **filename** (*str*) –
- **x** (*list or set (of kstFrozensets)*) –
- **Currently** –
- **files.** (*we write only classical KST format*) –

kstpy.simulation

Functions

simulate(structure, number, beta, gamma)	Simulate data from a structure according to the BLIM

Module Contents

kstpy.simulation.**simulate**(*structure*, *number*, *beta*, *gamma*)

Simulate data from a structure according to the BLIM

Parameters

- **structure** (*set or list*) – data basis for the simulation
- **number** (*int*) – number of response patterns to be simulated
- **beta** (*float*) – likelihood for careless errors
- **gamma** (*float*) – likelihood for lucky guesses

kstpy.validation

Functions

distvec(data, structure)	Compute a vector of distances from response patterns to a knowledge structure
difreq(data, structure)	Determine a vector of frequencies of distances between a set of response patterns and a knowledge structure
di(data, structure)	Determine the Discrepancy Index
da(data, structure)	Determine the Distance Agreement coefficient

Module Contents

kstpy.validation.**distvec**(*data*, *structure*)

Compute a vector of distances from response patterns to a knowledge structure

Parameters

- **data** (*list*) – List of kstFrozensets containing response patterns
- **structure** (*set*) – Family of kstFrozensets - the knowledge structure

Return type
Vector of distances between response patterns and knowledge structure

kstpy.validation.**difreq**(*data*, *structure*)

Determine a vector of frequencies of distances between a set of response patterns and a knowledge structure

Parameters

- **data** (*list*) – List of kstFrozensets containing response patterns
- **structure** (*set*) – Family of kstFrozensets - the knowledge structure

Return type
Vector of distance frequencies

kstpy.validation.**di**(*data*, *structure*)

Determine the Discrepancy Index

Parameters

structure: set
Family of kstFrozensets - the knowledge structure

Return type
Discrepancy Index

kstpy.validation.**da**(*data*, *structure*)

Determine the Distance Agreement coefficient

Parameters

- **data** (*list*) – List of kstFrozensets containing response patterns
- **structure** (*set*) – Family of kstFrozensets - the knowledge structure

Return type
Distance Agreement coefficient

Attributes

__version__

Package Contents

kstpy.__version__ = '0.1.1'

Python Module Index

k

Index

Symbols

B

C

D

E

F

G

H

I

K

L

M

N

P

R

S

V

W

X