Package ‘bayesvl’

July 22, 2025

Type Package

Title Visually Learning the Graphical Structure of Bayesian Networks and Performing MCMC with 'Stan'

Version 1.0.0

Date 2025-05-17

Author Viet-Phuong La [aut, cre],
Quan-Hoang Vuong [aut]

Maintainer Viet-Phuong La <lvphuong@gmail.com>

Imports coda, bnlearn, ggplot2, bayesplot, viridis, reshape2

Suggests loo (>= 2.0.0)

Depends R (>= 3.5.0), rstan (>= 2.10.0), StanHeaders (>= 2.18.0), stats, graphics, methods

Description Provides users with its associated functions for pedagogical purposes in visually learning Bayesian networks and Markov chain Monte Carlo (MCMC) computations. It enables users to: a) Create and examine the (starting) graphical structure of Bayesian networks; b) Create random Bayesian networks using a dataset with customized constraints; c) Generate Stan code for structures of Bayesian networks for sampling the data and learning parameters; d) Plot the network graphs; e) Perform Markov chain Monte Carlo computations and produce graphs for posteriors checks. The package refers to one reference item, which describes the methods and algorithms: Vuong, Quan-Hoang and La, Viet-Phuong (2019) <doi:10.31219/osf.io/w5dx6> The 'bayesvl' R package. Open Science Framework (May 18).

License GPL (>= 3)

BugReports https://github.com/sshpa/bayesvl/issues

URL https://github.com/sshpa/bayesvl

NeedsCompilation no

Repository CRAN

Date/Publication 2025-05-17 14:50:06 UTC

Contents

`bayesvl-package` *BayesVL: Visual Learning and Bayesian Statistical Analysis in R*

Description

An R package for visually constructing graphical models of Bayesian networks and performing Hamiltonian Monte Carlo (HMC) via Stan, using functions such as `bvl_model2Stan` and `bvl_modelFit`.

Details

Package:	bayesvl
Type:	Package
Version:	0.8.0
Date:	2019-05-13
License:	GPL-3
Website:	https://github.com/sshpa/bayesvl

Author(s)

Quan-Hoang Vuong, Viet-Phuong La

References

For documentation, case studies, worked examples, and other tutorial materials, visit the References section on our GitHub:

- https://github.com/sshpa/bayesvl/tree/master/References

For case studies using the package in research articles, see:

- https://www.nature.com/articles/s41599-020-0442-3

See Also

bayesvl-class, bvl_modelFit, bvl_model2Stan

Examples

```
# Create a new model
model <- bayesvl()

# Add observed data nodes
model <- bvl_addNode(model, "Lie", "binom")
model <- bvl_addNode(model, "B", "binom")
model <- bvl_addNode(model, "C", "binom")
model <- bvl_addNode(model, "T", "binom")

# Add directed arcs
model <- bvl_addArc(model, "B", "Lie", "slope")
model <- bvl_addArc(model, "C", "Lie", "slope")
model <- bvl_addArc(model, "T", "Lie", "slope")

# View model summary
summary(model)
```

`bayesvl bnlearn utilities` *bnlearn interface for bayesvl objects*

Description

Provides the interface to the functions in the bnlearn package for network diagnostics of an object of class `bayesvl`.

Usage

```
# Interface to bn.fit function to fit the parameters of
# a Bayesian network conditional on its structure.
bvl_bnBayes(dag, data = NULL, method = "bayes", iss = 10, ...)

# Interface to bnlearn score function to compute the score of the Bayesian network.
bvl_bnScore(dag, data = NULL, ...)

# Interface to arc.strength function to measure the strength of the probabilistic
# relationships expressed by the arcs of a Bayesian network.
bvl_bnStrength(dag, data = NULL, criterion = "x2", ...)

# Interface to bn.fit.barchart function to plot fit
# the parameters of a Bayesian network conditional on its structure.
bvl_bnBarchart(dag, data = NULL, method = "bayes", iss = 10, ...)
```

```
bvl_modelData (net, data)

bvl_compareLoo (dag1, dag2, ...)

bvl_compareWAIC (dag1, dag2, ...)
```

Arguments

`dag` an object of class `bayesvl`

`data` a data frame containing the variables in the model.

`method` a character string, either mle for Maximum Likelihood parameter estimation or bayes for Bayesian parameter estimation (currently implemented only for discrete data).

`iss` a numeric value, the imaginary sample size used by the bayes method to estimate the conditional probability tables associated with discrete nodes

`criterion` a character string, the method using for measuring

`net` network graph

`dag1` first model to compare

`dag2` second model to compare

`...` extra arguments from the generic method

Value

`bvl_bnScore()` return a number, value of score.

Author(s)

La Viet-Phuong, Vuong Quan-Hoang

References

For documentation, case studies, worked examples, and other tutorial materials, visit the References section on our GitHub:

- https://github.com/sshpa/bayesvl/tree/master/References

For case studies using the package in research articles, see:

- https://www.nature.com/articles/s41599-020-0442-3 bayesvl graph utilities

Utilities to manipulate graphs

Description

Manipulate directed acyclic graph of an object of class `bayesvl`.

Usage

```
# added a new node to the graph.
bvl_addNode(dag, name, dist = "norm", priors = NULL, fun = NULL, out_type = NULL,
  lower = NULL, upper=NULL, test = NULL)

# added a new path between nodes to the graph.
bvl_addArc(dag, from, to, type = "slope", priors = NULL, fun = NULL)

# added a new path between nodes to the graph.
bvl_addArc(dag, from, to, type = "slope", priors = NULL, fun = NULL)
```

Arguments

`dag`	an object of class `bayesvl`
`name`	a character string, the name of a node.
`dist`	a character string, distribution code of the node (`norm`, `binom`).
`priors`	a vector of string, the priors of the node or path.
`fun`	a character string, the transform function of the node.
`out_type`	a character string, the variable data type (`int`, `real`, ...).
`lower`	integer or real, the lower bound of variable data type (`int` or `real`).
`upper`	integer or real, the upper bound of variable data type (`int` or `real`).
`test`	a vector of testing values for variable.
`from`	a character string, the name of node the path connect from.
`to`	a character string, the name of node the path connect to.
`type`	a character string, the path type between nodes (`slope`, `varint`, ...).

Value

`bvl_addNode()`, `bvl_addArc()` return object class `bayesvl`.

Author(s)

La Viet-Phuong, Vuong Quan-Hoang

References

For documentation, case studies, worked examples, and other tutorial materials, visit the References section on our GitHub:

- https://github.com/sshpa/bayesvl/tree/master/References

For case studies using the package in research articles, see:

- https://www.nature.com/articles/s41599-020-0442-3

Examples

```
dag = bayesvl()

# add nodes to dag
dag = bvl_addNode(dag, "node1")
dag = bvl_addNode(dag, "node2")

# add the path between two nodes
dag = bvl_addArc(dag, "node1", "node2")

summary(dag)
```

---

bayesvl plot utilities

Plot utilities for bayesvl objects

Description

Provides plot methods and the interface to the MCMC module in the bayesplot package for plotting MCMC draws and diagnostics for an object of class bayesvl.

Usage

```
# Plot network diagram to visualize the model
bvl_bnPlot(dag, ...)

# Plots historgram of regression parameters computed from posterior draws in grid layout
bvl_plotParams (dag, row = 2, col = 2, credMass = 0.95, params = NULL)

# The interface to mcmc_intervals for plotting uncertainty intervals
# computed from posterior draws
bvl_plotIntervals (dag, params = NULL, fun = "mean", prob = 0.8,
  prob_outer = 0.95, color_scheme = "blue", labels = NULL)

# The interface to mcmc_intervals for plotting density computed from posterior draws
bvl_plotAreas (dag, params = NULL, fun = "mean",
  prob = 0.8, prob_outer = 0.95, color_scheme = "blue", labels = NULL)

bvl_plotPairs (dag, params = NULL, size = 1, color_scheme = "blue", labels = NULL)

bvl_plotDensity (dag, params = NULL, size = 1, labels = NULL)

bvl_plotDensity2d(dag, x, y, color = NULL, color_scheme = "red", labels = NULL)

bvl_plotTrace (dag, params = NULL)

bvl_plotDiag (dag)

bvl_plotGelman (dag, params = NULL)

bvl_plotGelmans (dag, params = NULL, row = 2, col = 2)

bvl_plotAc ( dag, params = NULL)

bvl_plotAcf ( dag, params = NULL)

bvl_plotAcfs ( dag, params = NULL, row = 2, col = 2)

bvl_plotAcf_Bar ( dag, params = NULL, color_scheme="pink",labels=NULL)

bvl_plotDensOverlay (dag, n = 200, color_scheme = "blue")

bvl_plotMCMCDiag ( dag, parName, saveName=NULL , saveType="jpg")

bvl_plotPPC (dag, fun = "stat", stat = "mean", color_scheme = "blue")

bvl_plotTest (dag, y_name, test_name, n = 200, color_scheme = "blue")
```

Arguments

`dag`	an object of class `bayesvl`
`params`	Optional: character vector of parameter names.
`fun`	Optional: statistic function.
`stat`	Optional: the plotting function to call.
`prob`	Optional: the probability mass to include in the inner interval. Default is 0.8.
`prob_outer`	Optional: the probability mass to include in the outer interval. Default is 0.95.
`row`	Optional: number of rows of grid layout.
`col`	Optional: number of columns of grid layout.
`credMass`	Optional: specifying the mass within the credible interval. Default is 0.89.
`size`	Optional: the size of line width.
`color_scheme`	Optional: color scheme. Default is "blue"
`...`	extra arguments from the generic method

`y_name`	a character string. Name of outcome variable
`test_name`	a character string. Name of test variable and test value
`n`	number of yrep values to plot
`x`	a character string. Name of x parameter to pair with
`y`	a character string. Name of y parameter to pair with
`color`	a character string. Variable for color of points on density plot
`labels`	Optional: character vector of parameter labels.
`parName`	parameter name for plotting.
`saveName`	file name for exporting plot.
`saveType`	type of file name for exporting plot (default is ’jpg’).

Value

`bvl_plotIntervals()`, `bvl_plotPairs()` return a ggplot object that can be further customized using the ggplot2 package.

Author(s)

La Viet-Phuong, Vuong Quan-Hoang

References

For documentation, case studies, worked examples, and other tutorial materials, visit the References section on our GitHub:

- https://github.com/sshpa/bayesvl/tree/master/References

For case studies using the package in research articles, see:

- https://www.nature.com/articles/s41599-020-0442-3

Examples

```
## create network model
model <- bayesvl()
## add the observed data nodes
model <- bvl_addNode(model, "O", "binom")
model <- bvl_addNode(model, "Lie", "binom")
model <- bvl_addNode(model, "Viol", "binom")
model <- bvl_addNode(model, "VB", "binom")
model <- bvl_addNode(model, "VC", "binom")
model <- bvl_addNode(model, "VT", "binom")
model <- bvl_addNode(model, "Int1", "binom")
model <- bvl_addNode(model, "Int2", "binom")

## add the tranform data nodes and arcs as part of the model
model <- bvl_addNode(model, "B_and_Viol", "trans")
model <- bvl_addNode(model, "C_and_Viol", "trans")
model <- bvl_addNode(model, "T_and_Viol", "trans")
model <- bvl_addArc(model, "VB",        "B_and_Viol", "*")
model <- bvl_addArc(model, "Viol",      "B_and_Viol", "*")
model <- bvl_addArc(model, "VC",        "C_and_Viol", "*")
model <- bvl_addArc(model, "Viol",      "C_and_Viol", "*")
model <- bvl_addArc(model, "VT",        "T_and_Viol", "*")
model <- bvl_addArc(model, "Viol",      "T_and_Viol", "*")
model <- bvl_addArc(model, "B_and_Viol",  "O", "slope")
model <- bvl_addArc(model, "C_and_Viol",  "O", "slope")
model <- bvl_addArc(model, "T_and_Viol",  "O", "slope")

model <- bvl_addArc(model, "Viol",   "O", "slope")

model <- bvl_addNode(model, "B_and_Lie", "trans")
model <- bvl_addNode(model, "C_and_Lie", "trans")
model <- bvl_addNode(model, "T_and_Lie", "trans")
model <- bvl_addArc(model, "VB",       "B_and_Lie", "*")
model <- bvl_addArc(model, "Lie",      "B_and_Lie", "*")
model <- bvl_addArc(model, "VC",       "C_and_Lie", "*")
model <- bvl_addArc(model, "Lie",      "C_and_Lie", "*")
model <- bvl_addArc(model, "VT",       "T_and_Lie", "*")
model <- bvl_addArc(model, "Lie",      "T_and_Lie", "*")
model <- bvl_addArc(model, "B_and_Lie",  "O", "slope")
model <- bvl_addArc(model, "C_and_Lie",  "O", "slope")
model <- bvl_addArc(model, "T_and_Lie",  "O", "slope")

model <- bvl_addArc(model, "Lie",   "O", "slope")

model <- bvl_addNode(model, "Int1_or_Int2", "trans")
model <- bvl_addArc(model, "Int1", "Int1_or_Int2", "+")
model <- bvl_addArc(model, "Int2", "Int1_or_Int2", "+")

model <- bvl_addArc(model, "Int1_or_Int2", "O", "varint")

## Plot network diagram to visualize the model
bvl_bnPlot(model)
```

`bayesvl-class` *Class* `bayesvl`*: Object Class for BayesVL Models*

Description

An S4 class that represents a Bayesian model created using the `bayesvl` package. This object is typically returned by functions such as `bayesvl`.

Slots

`call` Original function call that created the model.

nodes List of nodes in the model.

arcs List of arcs (edges) connecting the nodes.

pars List of model parameters.

stanfit An object of class stanfit, representing the fitted Stan model.

rawdata A data frame containing observed input data.

standata Data list used for Stan sampling.

posterior A data frame representation of posterior draws from the stanfit object.

elapsed Elapsed time for the MCMC simulation (in seconds).

Methods

show signature(object = "bayesvl"): Prints a default summary of the model.

summary Displays a more detailed overview of the model structure and output.

References

For documentation, case studies, worked examples, and other tutorial materials, visit our GitHub:

- https://github.com/sshpa/bayesvl/tree/master/References

For case studies using the package in research articles, refer to:

- https://www.nature.com/articles/s41599-020-0442-3

See Also

bayesvl

Examples

```
# Design the model in a directed acyclic graph
model <- bayesvl()

# Add observed data nodes to the model
model <- bvl_addNode(model, "Lie", "binom")
model <- bvl_addNode(model, "B", "binom")
model <- bvl_addNode(model, "C", "binom")
model <- bvl_addNode(model, "T", "binom")

# Add paths between nodes
model <- bvl_addArc(model, "B", "Lie", "slope")
model <- bvl_addArc(model, "C", "Lie", "slope")
model <- bvl_addArc(model, "T", "Lie", "slope")

# Summarize the model
summary(model)
```

`bayesvl-news` *News for Package 'bayesvl'*

Description

This page documents major changes and updates in the development of the **bayesvl** package.

Changes in version 1.0.0

- Updated many functions.
- Added posterior predictive check (PPC) support.

Changes in version 0.9.0

- Added WAIC estimation functions.
- Added LOO 2.0 estimation functions.
- Added model comparison functions.
- Updated `.Rd` documentation and other metadata.

Changes in version 0.8.5

- Updated `.Rd` documentation and other metadata.
- Fixed bugs for CRAN submission.

Changes in version 0.7.6

- Fixed error in single-node models.
- Updated `.Rd` documentation and other metadata.

Changes in version 0.7.0

- Fixed alpha intercept in varying intercept models.
- Fixed `lower = 0` constraint for varying intercept models.
- Renamed `net2stan.r` to `bayesvl2stan.r`.
- Added WAIC calculation support.

Changes in version 0.6.8

- Added arc templates.
- Added model validation functions.
- Added automatic generation of data list for Stan.
- Added log-likelihood comparison function.

Changes in version 0.6.5

- Supported node type `Dummy` for temporary parameters.
- Supported node type `Trans` for transformed data.
- Supported custom `generated quantities` block.
- Supported `y_rep` and `log_lik` output.
- Updated `README.md`.

Changes in version 0.6.0

- Added more distribution templates.
- Updated Stan code generator from network graph.
- Updated `README.md`.

Changes in version 0.5.1

- Numerous documentation updates.

Changes in version 0.5.0

- Added functions for Stan code generation.
- Added distribution templates.
- Updated `README.md`.

Changes in version 0.3.0

- Added `bnPlot()`, `bnScore()`, `bnStrength()` to interface with `bnlearn`.
- Added utilities to convert between bayesvl and bnlearn structures.
- Updated `README.md`.

Changes in version 0.2.0

- Added functions to add/remove nodes and arcs in the network graph.
- Added network initialization function.
- Implemented `bayesvl` class.
- First fully documented release.

Changes in version 0.1.0

- Initial package description and metadata.
- First public release.

bayesvl-stan-utilities *Build Stan Models from Directed Acyclic Graphs*

Description

Functions to generate Stan code and run simulations using a model object of class `bayesvl`, which represents a Bayesian directed acyclic graph (DAG).

Usage

```
bvl_model2Stan(dag, ppc = "")

bvl_modelFit(dag, data, warmup = 1000, iter = 5000, chains = 2, ppc = "", ...)

bvl_stanPriors(dag)

bvl_stanParams(dag)

bvl_formula (dag, nodeName, outcome = T, re = F)

bvl_stanLikelihood (dag)

bvl_stanLoo (dag, ...)

bvl_stanWAIC (dag, ...)
```

Arguments

`dag`	An object of class `bayesvl` representing the model DAG.
`data`	A data frame or list containing the observed data for model fitting.
`warmup`	Number of warmup iterations; defaults to half of `iter`.
`iter`	Total number of iterations for sampling. Default is 5000.
`chains`	Number of MCMC chains to run. Default is 2.
`ppc`	Optional: a character string containing Stan code for posterior predictive checks.
`...`	Additional arguments passed to underlying functions.
`nodeName`	The name of the node to generate formula for.
`outcome`	Logical. Whether to include outcome distribution. Default is `TRUE`.
`re`	Logical. Whether to recursively trace all upstream nodes. Default is `FALSE`.

Value

The following outputs are returned depending on the function used:

- `bvl_model2Stan`: Returns a character string containing the generated Stan model code.
- `bvl_modelFit`: Returns an object of class `bayesvl` with the following slots:
  - `model`: The Stan model code.
  - `stanfit`: A `stanfit` object returned by `rstan`.
  - `standata`: The data list used in sampling.
  - `pars`: A list of parameter names being monitored.
  - `formula`: The formula representation of the model.
- `bvl_stanPriors`: Returns a character string of the prior distributions used in the model.
- `bvl_stanParams`: Returns a character string of parameter block content for Stan.
- `bvl_formula`: Returns the formula associated with the specified node.

Author(s)

La Viet-Phuong, Vuong Quan-Hoang

References

For documentation, case studies, worked examples, and other tutorial materials, see:

- User Guide and Examples on GitHub
- Published Case Study in Humanities and Social Sciences Communications

Examples

```
# Design the model using a directed acyclic graph
model <- bayesvl()
model <- bvl_addNode(model, "Lie", "binom")
model <- bvl_addNode(model, "B", "binom")
model <- bvl_addNode(model, "C", "binom")
model <- bvl_addNode(model, "T", "binom")

model <- bvl_addArc(model, "B", "Lie", "slope")
model <- bvl_addArc(model, "C", "Lie", "slope")
model <- bvl_addArc(model, "T", "Lie", "slope")

# Generate Stan model code
model_string <- bvl_model2Stan(model)
cat(model_string)

# Display priors in generated Stan model
bvl_stanPriors(model)
```

`data1042` *Dataset on Health, Insurance, and Financial Destitution in Vietnam*

Description

A dataset of 1,042 inpatients from hospitals in Northern Vietnam, collected over 20 months (August 2014 – March 2016). The dataset covers healthcare access, health insurance, treatment costs, financial burden, and socio-demographic variables. It has been used in multiple peer-reviewed publications.

Usage

```
data(data1042)
```

Format

A data frame with 1,042 observations and 45 variables. Selected variables:

`Age` Patient’s age.

`Burden` Financial burden after treatment.

`Days` Length of hospital stay.

`Dcost` Daily hospital cost.

`Edu` Educational attainment.

`End` Treatment outcome.

`IfHigher` Expected financial impact if treatment continued.

`Illness` Severity/type of illness.

`Income` Annual income.

`Insured` Whether the patient had insurance.

`Pchar` Portion covered by charity.

`Pinc` Portion covered by income.

`Pins` Portion covered by insurance reimbursement.

`Ploan` Portion covered by loans.

`Res` Region of residence.

`SES` Socioeconomic status.

`SatIns` Satisfaction with insurance.

`Saving` Percentage of savings used.

`Sex` Patient’s gender (1 = female, 2 = male).

`Spent` Total amount spent on treatment.

`AvgCost` Not yet documented.

`Dcost_USD` Not yet documented.

`EnvL` Not yet documented.

`Hospital` Not yet documented.

`ID` Not yet documented.

`Ill2` Not yet documented.

`IncRank` Not yet documented.

`Income_USD` Not yet documented.

`InsGap` Not yet documented.

`InsL` Not yet documented.

`InsL2` Not yet documented.

`Jcond` Not yet documented.

`LoanL` Not yet documented.

`MaxIns` Not yet documented.

`SatServ` Not yet documented.

`Senv` Not yet documented.

`Spent_USD` Not yet documented.

`Srel` Not yet documented.

`Stay` Not yet documented.

`Streat` Not yet documented.

`WkYrs` Not yet documented.

References

Ho, M.T.; La, V.P.; Nguyen, M.H.; Vuong, Q.H. et al. (2019). "Health care, health insurance and economic destitution: A dataset of 1042 stories." *Data*, 4.
https://www.mdpi.com/journal/data

Related studies:

- Vuong, Q.H. (2015). Be rich or don't be sick. *SpringerPlus*. doi:10.1186/s400640151279x
- Vuong, Q.H. (2016). Data on Vietnamese patients' financial burdens. *Data in Brief*. doi:10.1016/j.dib.2016.09.040
- Vuong, Q.H. (2017). Health insurance thresholds in Vietnam. *Biomedical Research*.

Examples

```
data(data1042)

# View structure
str(data1042)

# Summarize financial burden
table(data1042$Burden)
```

`DKAP1061` *DKAP1061 Dataset*

Description

DKAP1061 is a dataset from a survey on students' digital competence, including demographic and educational background variables.

Usage

```
data(DKAP1061)
```

Format

A data frame with multiple columns. Selected variables:

`ecostt` Student's family economic status.

`edufat` Father's education level.

`edumot` Mother's education level.

`ict` Digital competence score.

`mean_dr` Mean digital resources.

`mean_ict` Mean ICT skills score.

`mean_il` Mean information literacy score.

`mean_ppr` Mean personal productivity rating.

`mean_udcr` Mean use of digital content/resources.

`schoolid` School ID.

`schid` School code (alternative to schoolid).

`sex` Student's gender (1 = female, 2 = male).

`stuid` Student ID.

`a1` Not yet documented.

`a10` Not yet documented.

`a11` Not yet documented.

`a12` Not yet documented.

`a13` Not yet documented.

`a14` Not yet documented.

`a2` Not yet documented.

`a3` Not yet documented.

`a4` Not yet documented.

`a5` Not yet documented.

`a6` Not yet documented.

a7 Not yet documented.

a8 Not yet documented.

a9 Not yet documented.

b1 Not yet documented.

b10 Not yet documented.

b11 Not yet documented.

b12 Not yet documented.

b13 Not yet documented.

b14 Not yet documented.

b15_1 Not yet documented.

b15_2 Not yet documented.

b15_3 Not yet documented.

b15_4 Not yet documented.

b15_5 Not yet documented.

b15_6 Not yet documented.

b15_7 Not yet documented.

b15_8 Not yet documented.

b16_1 Not yet documented.

b16_2 Not yet documented.

b16_3 Not yet documented.

b16_4 Not yet documented.

b16_5 Not yet documented.

b16_6 Not yet documented.

b16_7 Not yet documented.

b16_8 Not yet documented.

b17_1 Not yet documented.

b17_2 Not yet documented.

b17_3 Not yet documented.

b17_4 Not yet documented.

b17_5 Not yet documented.

b17_6 Not yet documented.

b17_7 Not yet documented.

b17_8 Not yet documented.

b18_1 Not yet documented.

b18_2 Not yet documented.

b18_3 Not yet documented.

b18_4 Not yet documented.

b18_5 Not yet documented.

b18_6 Not yet documented.

b18_7 Not yet documented.

b18_8 Not yet documented.

b18_9 Not yet documented.

b2 Not yet documented.

b3 Not yet documented.

b4 Not yet documented.

b5 Not yet documented.

b6 Not yet documented.

b7 Not yet documented.

b8 Not yet documented.

b9 Not yet documented.

c1 Not yet documented.

c10 Not yet documented.

c11 Not yet documented.

c12 Not yet documented.

c2 Not yet documented.

c3 Not yet documented.

c4 Not yet documented.

c5 Not yet documented.

c6 Not yet documented.

c7 Not yet documented.

c8 Not yet documented.

c9 Not yet documented.

d1 Not yet documented.

d10 Not yet documented.

d11 Not yet documented.

d12 Not yet documented.

d13 Not yet documented.

d14 Not yet documented.

d15 Not yet documented.

d16 Not yet documented.

d2 Not yet documented.

d3 Not yet documented.

d4 Not yet documented.

d5 Not yet documented.

`d6` Not yet documented.

`d7` Not yet documented.

`d8` Not yet documented.

`d9` Not yet documented.

`e1` Not yet documented.

`e10` Not yet documented.

`e11` Not yet documented.

`e2` Not yet documented.

`e3` Not yet documented.

`e4` Not yet documented.

`e5` Not yet documented.

`e6` Not yet documented.

`e7` Not yet documented.

`e8` Not yet documented.

`e9` Not yet documented.

`f1` Not yet documented.

`f2` Not yet documented.

`f3` Not yet documented.

`f4` Not yet documented.

`f5` Not yet documented.

`f6` Not yet documented.

`f7` Not yet documented.

`f8_1` Not yet documented.

`f8_2` Not yet documented.

`f8_3` Not yet documented.

`f8_4` Not yet documented.

`f8_5` Not yet documented.

`g1` Not yet documented.

`g10_1` Not yet documented.

`g10_2` Not yet documented.

`g10_3` Not yet documented.

`g11` Not yet documented.

`g12` Not yet documented.

`g13` Not yet documented.

`g14` Not yet documented.

`g15` Not yet documented.

`g16` Not yet documented.

g17 Not yet documented.

g18 Not yet documented.

g2 Not yet documented.

g3_1 Not yet documented.

g3_2 Not yet documented.

g3_3 Not yet documented.

g3_4 Not yet documented.

g4_1 Not yet documented.

g4_2 Not yet documented.

g4_3 Not yet documented.

g4_4 Not yet documented.

g4_5 Not yet documented.

g4_6 Not yet documented.

g5_1 Not yet documented.

g5_2 Not yet documented.

g5_3 Not yet documented.

g5_4 Not yet documented.

g5_5 Not yet documented.

g5_6 Not yet documented.

g6_1 Not yet documented.

g6_2 Not yet documented.

g6_3 Not yet documented.

g6_4 Not yet documented.

g6_5 Not yet documented.

g6_6 Not yet documented.

g7_1 Not yet documented.

g7_2 Not yet documented.

g7_3 Not yet documented.

g8_1 Not yet documented.

g8_2 Not yet documented.

g8_3 Not yet documented.

g9 Not yet documented.

h1_1 Not yet documented.

h1_2 Not yet documented.

h1_3 Not yet documented.

h1_4 Not yet documented.

h1_5 Not yet documented.

h1_6 Not yet documented.

h1_7 Not yet documented.

h2 Not yet documented.

h3 Not yet documented.

h4_1 Not yet documented.

h4_2 Not yet documented.

h4_3 Not yet documented.

h5 Not yet documented.

h6_1 Not yet documented.

h6_2 Not yet documented.

h6_3 Not yet documented.

h6_4 Not yet documented.

h7_1 Not yet documented.

h7_2 Not yet documented.

h7_3 Not yet documented.

h7_4 Not yet documented.

Note: Variables starting with a, b, c, d, f, g, h *are omitted from this documentation.*

References

For documentation, case studies, and examples, visit the GitHub repository:

- Documentation and User Guide
- Published Case Study (Vuong & La, 2020)

Examples

```
data(DKAP1061)

# Preview the dataset
head(DKAP1061)
```

Legends345 *Legends345 data*

Description

Legends345.

Usage

```
data(Legends345)
```

Format

1. O : Whether or not happy ending for main character
2. VB : Whether or not the main character behaves in accordance with the core values of Buddhism
3. VC : Whether or not the main character behaves in accordance with the core values of Confucianism
4. VT : Whether or not the main character behaves in accordance with the core values of Taoism
5. Lie : Whether or not the main character tells lie
6. Viol : Whether or not the main character commits acts of violence
7. Int1 : Whether there are interventions from the supernatural world
8. Int2 : Whether there are interventions from the human world

References

For documentation, case studies, worked examples, and other tutorial materials, visit the References section on our GitHub:

- https://github.com/sshpa/bayesvl/tree/master/References

For case studies using the package in research articles, see:

- https://www.nature.com/articles/s41599-020-0442-3

Examples

```
data(Legends345)

data1 <- Legends345
head(data1)
```

Index