Package 'svines'

August 31, 2026

Title Stationary Vine Copula Models

Version 0.3.0

Description Provides functionality to fit and simulate from stationary vine copula models for time series, see Nagler et al. (2022) <doi:10.1016/j.jeconom.2021.11.015>.

License GPL-3

Encoding UTF-8

LazyData true

URL https://tnagler.github.io/svines/, https://github.com/tnagler/svines

BugReports https://github.com/tnagler/svines/issues

SystemRequirements C++17

Depends R (>= 4.3.0), rvinecopulib (>= 0.7.2.1.0)

Imports Rcpp, assertthat, univariateML, wdm, fGarch

LinkingTo RcppEigen, Rcpp, RcppThread, BH, wdm, rvinecopulib

Suggests testthat, ggraph, covr, knitr, rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/Needs/website pkgdown

NeedsCompilation yes

Author Thomas Nagler [aut, cre]

Maintainer Thomas Nagler <mail@tnagler.com>

Repository CRAN

Date/Publication 2026-08-31 21:50:08 UTC

Contents

`returns` *Stock returns of 20 companies*

Description

Daily log returns of 20 companies from the insurance, automotive, technology, energy, and aviation sectors. The observation period is from 2015-01-01 to 2019-12-31. Trading dates are not stored in the dataset.

Usage

```
returns
```

Format

A data frame with 1,296 observations and 20 numeric variables, one for each company: Allianz, AXA, Generali, MetLife, Prudential, Ping An, BMW, General Motors, Toyota, Hyundai, Microsoft, Apple, Amazon, Alphabet, Alibaba, Exxon, Shell, PetroChina, Airbus, and Boeing.

Source

Historical prices obtained from Yahoo Finance.

svine *Stationary vine distribution models*

Description

Automated fitting or creation of custom S-vine distribution models

Usage

```
svine(
  data,
  p,
  var_types,
  margin_families = univariateML::univariateML_models,
  selcrit = "aic",
  ...
)
```

Arguments

`data`	a matrix or data.frame of data.
`p`	the Markov order.
`var_types`	variable types; a character vector with one entry per variable: `"c"` for continuous, `"d"` for discrete. Defaults to all-continuous when missing.
`margin_families`	either a character vector of univariateML::univariateML_models to select from for every margin, or a list with one entry for every variable. For a discrete variable, the corresponding entry must contain only suitable discrete families. Use `"empirical"` for empirical CDFs.
`selcrit`	criterion for family selection, either `"loglik"`, `"aic"`, `"bic"`, `"mbicv"`.
`...`	arguments passed to `svinecop()`.

Value

Returns the fitted model as an object with classes `svine` and svine_dist. A list with entries

- `$margins`: list of marginal models from univariateML::univariateML_models,
- `$copula`: an object of `svinecop_dist`.

References

Nagler, T., Krüger, D., and Min, A. (2022). Stationary vine copula models for multivariate time series. *Journal of Econometrics*, 227(2), 305–324. doi:10.1016/j.jeconom.2021.11.015.

See Also

svine_dist, svine_loglik, svine_sim, svine_bootstrap_models

Examples

```
# load data set
data(returns)

# fit parametric S-vine model with Markov order 1
fit <- svine(returns[1:100, 1:3], p = 1, family_set = "parametric")
fit
summary(fit)
plot(fit$copula)
contour(fit$copula)
logLik(fit)

pairs(svine_sim(500, rep = 1, fit))
```

---

svinecop *Stationary vine copula models*

Description

Automated fitting or creation of custom S-vine copula models

Usage

```
svinecop(
  data,
  p,
  var_types,
  family_set = "all",
  cs_structure = NA,
  out_vertices = NA,
  in_vertices = NA,
  type = "S",
  par_method = "mle",
  nonpar_method = "constant",
  mult = 1,
  selcrit = "aic",
  weights = numeric(),
  psi0 = 0.9,
  presel = TRUE,
  trunc_lvl = Inf,
  tree_crit = "tau",
  threshold = 0,
  keep_data = FALSE,
  show_trace = FALSE,
  cores = 1
)
```

Arguments

data	a matrix or data.frame of pseudo-observations (approximately uniform margins). For discrete variables declared in var_types, append one left-limit column F(x-) per discrete variable after all regular columns. Use svine() to handle this transformation automatically.
p	the Markov order.
var_types	variable types; a character vector with one entry per variable: "c" for continuous, "d" for discrete. Omitting var_types when cs_structure is not provided will silently fit a continuous model.
family_set	a character vector of families; see rvinecopulib::bicop() for additional options.
cs_structure	the cross-sectional vine structure (see rvinecopulib::rvine_structure()); cs_structure = NA performs automatic structure selection.
out_vertices	a permutation of 1, ..., d specifying the out-vertices. It is selected automatically when no structure is provided. A fixed cs_structure requires explicit out_vertices and in_vertices.
in_vertices	a permutation of 1, ..., d specifying the in-vertices. It is selected automatically when no structure is provided. A fixed cs_structure requires explicit out_vertices and in_vertices.
type	type of stationary vine; "S" (default) for general S-vines, "D" for Smith's long D-vine, "M" for Beare and Seo's M-vine.
par_method	the estimation method for parametric models, either "mle" for sequential maximum likelihood, "itau" for inversion of Kendall's tau (only available for one-parameter families and "t").
nonpar_method	the estimation method for nonparametric models, either "constant" for the standard transformation estimator, or "linear"/"quadratic" for the local-likelihood approximations of order one/two.
mult	multiplier for the smoothing parameters of nonparametric families. Values larger than 1 make the estimate more smooth, values less than 1 less smooth.
selcrit	criterion for family selection, either "loglik", "aic", "bic", "mbic", or "mbicv".
weights	optional vector of weights for each observation.
psi0	prior probability of a non-independence copula (only used for selcrit = "mbic" and selcrit = "mbicv").
presel	whether the family set should be thinned out according to symmetry characteristics of the data.
trunc_lvl	the truncation level. Only Inf, corresponding to no truncation, is currently supported.
tree_crit	the criterion for tree selection, one of "tau", "rho", "hoeffd", or "mcor" for Kendall's τ, Spearman's ρ, Hoeffding's D, and maximum correlation, respectively.
threshold	for thresholded vine copulas; NA indicates that the threshold should be selected automatically by rvinecopulib::mBICV().
keep_data	whether the data should be stored (necessary for using fitted()).

show_trace	logical; whether a trace of the fitting progress should be printed.
cores	number of cores to use; if more than 1, estimation of pair copulas within a tree is done in parallel.

Value

Returns the fitted model as an object with classes svinecop and svinecop_dist. Also inherits from vinecop, vinecop_dist such that many functions from rvinecopulib can be called.

Examples

```
# load data set
data(returns)

# convert to pseudo observations with empirical cdf for marginal distributions
u <- pseudo_obs(returns[1:100, 1:3])

# fit parametric S-vine copula model with Markov order 1
fit <- svinecop(u, p = 1, family_set = "parametric")
fit
summary(fit)
plot(fit)
contour(fit)
logLik(fit)

pairs(svinecop_sim(500, rep = 1, fit))
```

---

svinecop_dist *Custom S-vine models*

Description

Custom S-vine models

Usage

```
svinecop_dist(
  pair_copulas,
  cs_structure,
  p,
  out_vertices,
  in_vertices,
  var_types = rep("c", dim(cs_structure)[1])
)
```

Arguments

pair_copulas	A nested list of 'bicop_dist' objects, where pair_copulas[[t]][[e]] corresponds to the pair-copula at edge e in tree t. Only the most-left unique pair copulas are used, others can be omitted.
cs_structure	The cross-sectional structure. Either a matrix, or an rvine_structure object; see rvinecopulib::rvine_structure()
p	the Markov order.
out_vertices	a permutation of 1, ..., d specifying the out-vertices.
in_vertices	a permutation of 1, ..., d specifying the in-vertices.
var_types	variable types; a character vector with one entry per variable: "c" for continuous, "d" for discrete. For discrete variables, methods on the returned object expect data in the compact layout: all regular CDF F(x) columns first, then one left-limit column F(x-) per discrete variable.

Value

Returns the model as an object with classes svinecop_dist. Also inherits from vinecop_dist such that many functions from rvinecopulib can be called.

See Also

svinecop_loglik, svinecop_sim, svinecop_hessian, svinecop_scores

Examples

```
cs_struct <- cvine_structure(1:2)
pcs <- list(
  list(  # first tree
    bicop_dist("clayton", 0, 3), # cross sectional copula
    bicop_dist("gaussian", 0, -0.1)  # serial copula
  ),
  list(  # second tree
    bicop_dist("gaussian", 0, 0.2), bicop_dist("indep")
  ),
  list( # third tree
    bicop_dist("indep")
  )
)

cop <- svinecop_dist(
  pcs, cs_struct, p = 1, out_vertices = 1:2, in_vertices = 1:2)
```

svinecop_hessian *Expected Hessian for S-vine copula models*

Description

Expected Hessian for S-vine copula models

Usage

```
svinecop_hessian(u, model, cores = 1)
```

Arguments

u	the data; should have approximately uniform margins.
model	model inheriting from class svinecop_dist.
cores	number of cores to use; if larger than one, computations are performed in parallel on cores batches.

Value

A model$npars-by-model$npars estimate of the expected Hessian, obtained by averaging the per-observation Hessian contributions. Rows and columns correspond to model parameters in the order: copula parameters of the first tree, copula parameters of the second tree, etc. Duplicated parameters in the copula model are omitted.

See Also

svinecop_scores

Examples

```
# load data set
data(returns)

# convert to uniform margins
u <- pseudo_obs(returns[1:100, 1:3])

# fit parametric S-vine copula model with Markov order 1
fit <- svinecop(u, p = 1, family_set = "parametric")

svinecop_loglik(u, fit)
svinecop_scores(u, fit)
svinecop_hessian(u, fit)
```

svinecop_loglik *Log-likelihood for S-vine copula models*

Description

Log-likelihood for S-vine copula models

Usage

```
svinecop_loglik(u, model, cores = 1)
```

Arguments

u	the data; should have approximately uniform margins.
model	model inheriting from class svinecop_dist.
cores	number of cores to use; if larger than one, computations are performed in parallel on cores batches.

Value

The log-likelihood of the data under the model.

Examples

```
# load data set
data(returns)

# convert to uniform margins
u <- pseudo_obs(returns[1:100, 1:3])

# fit parametric S-vine copula model with Markov order 1
fit <- svinecop(u, p = 1, family_set = "parametric")

svinecop_loglik(u, fit)
svinecop_scores(u, fit)
svinecop_hessian(u, fit)
```

svinecop_pseudo_residuals *Pseudo-residuals of S-vine copula models*

Description

Pseudo-residuals are defined as the Rosenblatt transform of the data, conditional on the past. Under a correctly specified model, they are approximately *iid* uniform on $[0, 1]^d$.

Usage

```
svinecop_pseudo_residuals(u, model, cores = 1)
```

Arguments

u	the data; should have approximately uniform margins.
model	model inheriting from class svinecop_dist.
cores	number of cores to use; if larger than one, computations are performed in parallel on cores batches.

Value

An n-by-d matrix of pseudo-residuals, where n = NROW(u) - model$p and d is the cross-sectional dimension.

Examples

```
# load data set
data(returns)

# convert to pseudo observations with empirical cdf for marginal distributions
u <- pseudo_obs(returns[1:100, 1:3])

# fit parametric S-vine copula model with Markov order 1
fit <- svinecop(u, p = 1, family_set = "parametric")

# compute pseudo-residuals
# (should be independent uniform across variables and time)
v <- svinecop_pseudo_residuals(u, fit)
pairs(cbind(v[-1, ], v[-nrow(v), ]))
```

---

svinecop_scores *Log-likelihood scores for S-vine copula models*

Description

Log-likelihood scores for S-vine copula models

Usage

```
svinecop_scores(u, model, cores = 1)
```

Arguments

u	the data; should have approximately uniform margins.
model	model inheriting from class svinecop_dist.
cores	number of cores to use; if larger than one, computations are performed in parallel on cores batches.

Value

An n-by-k matrix containing the score vectors in its rows, where n = NROW(u) - model$p and k = model$npars. The rows correspond to the effective time points after the first p observations. The columns correspond to model parameters in the order: copula parameters of first tree, copula parameters of second tree, etc. Duplicated parameters in the copula model are omitted.

See Also

svinecop_hessian

Examples

```
# load data set
data(returns)

# convert to uniform margins
u <- pseudo_obs(returns[1:100, 1:3])

# fit parametric S-vine copula model with Markov order 1
fit <- svinecop(u, p = 1, family_set = "parametric")

svinecop_loglik(u, fit)
svinecop_scores(u, fit)
svinecop_hessian(u, fit)
```

---

svinecop_sim *Simulate from a S-vine copula model*

Description

Simulate from a S-vine copula model

Usage

```
svinecop_sim(n, rep, model, past = NULL, qrng = FALSE, cores = 1)
```

Arguments

n	how many steps of the time series to simulate.
rep	number of replications; rep time series of length n are generated.
model	a S-vine copula model object (inheriting from svinecop_dist).
past	(optional) matrix of past observations. If provided, time series are simulated conditional on the past.
qrng	if TRUE, generates quasi-random numbers using the multivariate Generalized Halton sequence up to dimension 300 and the Generalized Sobol sequence in higher dimensions (default qrng = FALSE).
cores	number of cores to use; if larger than one, computations are performed in parallel over replications.

Value

An n-by-d-by-rep array, where d is the cross-sectional dimension of the model. This reduces to an n-by-d matrix if rep == 1.

Examples

```
# load data set
data(returns)

# convert to uniform margins
u <- pseudo_obs(returns[1:100, 1:3])

# fit parametric S-vine copula model with Markov order 1
fit <- svinecop(u, p = 1, family_set = "parametric")

pairs(u)   # original data
pairs(svinecop_sim(100, rep = 1, model = fit))   # simulated data

# simulate the next day conditionally on the past 500 times
pairs(t(svinecop_sim(1, rep = 100, model = fit, past = u)[1, , ]))
```

---

svine_bootstrap_models

Bootstrap S-vine models

Description

Computes bootstrap replicates of a given model using the one-step block multiplier bootstrap of Nagler et al. (2022).

Usage

```
svine_bootstrap_models(n_models, model)
```

Arguments

n_models	number of bootstrap replicates.
model	the initial fitted model

Value

A list of length n_models, with each entry representing one bootstrapped model as object of class svine.

References

Nagler, T., Krüger, D., and Min, A. (2022). Stationary vine copula models for multivariate time series. *Journal of Econometrics*, 227(2), 305–324. doi:10.1016/j.jeconom.2021.11.015.

Examples

```
data(returns)
dat <- returns[1:100, 1:2]

# fit parametric S-vine model with Markov order 1
model <- svine(dat, p = 1, family_set = "parametric")

# compute 10 bootstrap replicates of the model
boot_models <- svine_bootstrap_models(10, model)

# compute bootstrap replicates of 90%-quantile of X_1 + X_2.
mu_boot <- sapply(
  boot_models,
  function(m) {
    xx <- rowSums(t(svine_sim(1, 10^2, m, past = dat)[1, ,]))
    quantile(xx, 0.9)
  }
)
```

---

svine_dist *Custom S-vine distribution models*

Description

Custom S-vine distribution models

Usage

```
svine_dist(margins, copula)
```

Arguments

margins	A list with one marginal distribution per variable. Each element is either a univariateML object or a list with callables $p (CDF), $q (quantile function), and for discrete margins $p_sub (left-limit CDF, i.e. F(x-)). Discrete list-form margins must also carry attr(., "type") = "discrete" and class including "svine_margin".
copula	the copula model; an object of class svinecop_dist with cross-sectional dimension d.

Value

Returns the model as an object with class svine_dist. A list with entries

- $margins: list of marginal models from univariateML::univariateML_models,
- $copula: an object of svinecop_dist.

See Also

svine_dist, svine_loglik, svine_sim, svine_bootstrap_models

Examples

```
## marginal objects
# create dummy univariateML models
univ1 <- univ2 <- univariateML::mlnorm(rnorm(10))

# modify the parameters to N(5, 10) and N(0, 2) distributions
univ1[] <- c(5, 10)
univ2[] <- c(0, 2)

## copula object
cs_struct <- cvine_structure(1:2)
pcs <- list(
  list(  # first tree
    bicop_dist("clayton", 0, 3), # cross sectional copula
    bicop_dist("gaussian", 0, -0.1)  # serial copula
  ),
  list(  # second tree
    bicop_dist("gaussian", 0, 0.2), bicop_dist("indep")
  ),
  list( # third tree
    bicop_dist("indep")
  )
)

cop <- svinecop_dist(
  pcs, cs_struct, p = 1, out_vertices = 1:2, in_vertices = 1:2)

model <- svine_dist(margins = list(univ1, univ2), copula = cop)
summary(model)
```

---

svine_hessian *Expected Hessian of a parametric S-vine model*

Description

Expected Hessian of a parametric S-vine model

Usage

```
svine_hessian(x, model, cores = 1)
```

Arguments

x	the data.
model	S-vine model (inheriting from svine_dist).
cores	number of cores to use.

Value

A k-by-k estimate of the expected Hessian, where k is the total number of model parameters. The estimate averages the per-observation Hessian contributions. Parameters are ordered as follows: marginal parameters, copula parameters of first tree, copula parameters of second tree, etc. Duplicated parameters in the copula model are omitted.

Examples

```
data(returns)
dat <- returns[1:100, 1:2]

# fit parametric S-vine model with Markov order 1
model <- svine(dat, p = 1, family_set = "parametric")

# Implementation of asymptotic variances
I <- cov(svine_scores(dat, model))
H <- svine_hessian(dat, model)
Hi <- solve(H)
n_eff <- nrow(dat) - model$copula$p
Hi %*% I %*% t(Hi) / n_eff
```

---

svine_loglik *Log-likelihood for S-vine models*

Description

Log-likelihood for S-vine models

Usage

```
svine_loglik(x, model, cores = 1)
```

Arguments

x	the data.
model	model inheriting from class svine_dist.
cores	number of cores to use; if larger than one, computations are performed in parallel on cores batches.

Value

The log-likelihood of the data under the model.

Examples

```
# load data set
data(returns)

# fit parametric S-vine model with Markov order 1
fit <- svine(returns[1:100, 1:3], p = 1, family_set = "parametric")

svine_loglik(returns[1:100, 1:3], fit)
```

`svine_pseudo_residuals` *Pseudo-residuals of S-vine models*

Description

Pseudo-residuals are defined as the Rosenblatt transform of the data, conditional on the past. Under a correctly specified model, they are approximately *iid* uniform on $[0, 1]^d$.

Usage

```
svine_pseudo_residuals(x, model, cores = 1)
```

Arguments

`x`	the data.
`model`	model inheriting from class svine_dist.
`cores`	number of cores to use; if larger than one, computations are performed in parallel on `cores` batches.

Value

An `n`-by-`d` matrix of pseudo-residuals, where `n = NROW(x) - model$copula$p` and `d` is the cross-sectional dimension.

Examples

```
# load data set
data(returns)

dat <- returns[1:100, 1:3]

# fit parametric S-vine model with Markov order 1
fit <- svine(dat, p = 1, family_set = "parametric")

# compute pseudo-residuals
# (should be independent uniform across variables and time)
v <- svine_pseudo_residuals(dat, fit)
pairs(cbind(v[-1, ], v[-nrow(v), ]))
```

svine_scores *Score function of parametric S-vine models*

Description

Score function of parametric S-vine models

Usage

```
svine_scores(x, model, cores = 1)
```

Arguments

x	the data.
model	S-vine model (inheriting from svine_dist).
cores	number of cores to use.

Value

An n-by-k matrix, where n = NROW(x) - model$copula$p and k is the total number of model parameters. The rows correspond to the effective time points after the first p observations. Parameters are ordered as follows: marginal parameters, copula parameters of first tree, copula parameters of second tree, etc. Duplicated parameters in the copula model are omitted.

Examples

```
data(returns)
dat <- returns[1:100, 1:2]

# fit parametric S-vine model with Markov order 1
model <- svine(dat, p = 1, family_set = "parametric")

# Implementation of asymptotic variances
I <- cov(svine_scores(dat, model))
H <- svine_hessian(dat, model)
Hi <- solve(H)
n_eff <- nrow(dat) - model$copula$p
Hi %*% I %*% t(Hi) / n_eff
```

svine_sim *Simulate from a S-vine model*

Description

Simulate from a S-vine model

Usage

```
svine_sim(n, rep, model, past = NULL, qrng = FALSE, cores = 1)
```

Arguments

n	how many steps of the time series to simulate.
rep	number of replications; rep time series of length n are generated.
model	a S-vine copula model object (inheriting from svinecop_dist).
past	(optional) matrix of past observations. If provided, time series are simulated conditional on the past.
qrng	if TRUE, generates quasi-random numbers using the multivariate Generalized Halton sequence up to dimension 300 and the Generalized Sobol sequence in higher dimensions (default qrng = FALSE).
cores	number of cores to use; if larger than one, computations are performed in parallel over replications.

Value

An n-by-d-byrep array, where d is the cross-sectional dimension of the model. This reduces to an n-by-d matrix if rep == 1.

Examples

```
# load data set
data(returns)
returns <- returns[1:100, 1:3]

# fit parametric S-vine model with Markov order 1
fit <- svine(returns, p = 1, family_set = "parametric")

pairs(returns)  # original data
pairs(svine_sim(100, rep = 1, model = fit))   # simulated data

# simulate the next day conditionally on the past 500 times
pairs(t(svine_sim(1, rep = 100, model = fit, past = returns)[1, , ]))
```

Index