

# Package: svines (via r-universe)

September 18, 2024

**Title** Stationary Vine Copula Models

**Version** 0.2.3

**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://github.com/tnagler/svines>

**BugReports** <https://github.com/tnagler/svines/issues>

**Depends** R (>= 4.0.0), rvinecopulib (>= 0.6.1.1.2)

**Imports** Rcpp, assertthat, univariateML, wdm, fGarch

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

**Suggests** testthat, ggraph, covr

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Repository** <https://tnagler.r-universe.dev>

**RemoteUrl** <https://github.com/tnagler/svines>

**RemoteRef** HEAD

**RemoteSha** 28c91fc1b0dcb816557dd3a2efb9524cdddd8b11

## Contents

returns . . . . .	2
svine . . . . .	2
svinecop . . . . .	3
svinecop_dist . . . . .	6
svinecop_hessian . . . . .	7
svinecop_loglik . . . . .	8

svinecop_pseudo_residuals . . . . .	9
svinecop_scores . . . . .	10
svinecop_sim . . . . .	11
svine_bootstrap_models . . . . .	12
svine_dist . . . . .	13
svine_hessian . . . . .	14
svine_loglik . . . . .	15
svine_pseudo_residuals . . . . .	15
svine_scores . . . . .	16
svine_sim . . . . .	17

## Index 19

---

returns	<i>Stock returns of 20 companies</i>
---------	--------------------------------------

---

### Description

A dataset containing the log-returns of daily returns of 20 companies. The observation period is from 2015-01-01 to 2019-12-31.

### Usage

returns

### Format

A data frame with 1296 rows and 20 variables:

### Source

Yahoo finance.

---

svine	<i>Stationary vine distribution models</i>
-------	--

---

### Description

Automated fitting or creation of custom S-vine distribution models

### Usage

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

**Arguments**

data	a matrix or data.frame of data.
p	the Markov order.
margin_families	either a vector of <a href="#">univariateML</a> families to select from (used for every margin) or a list with one entry for every variable. Can also be "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

- `$margins`: list of marginal models from [univariateML](#),
- `$copula`: an object of `svinecop_dist`.

**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 = rep("c", NCOL(data)),
  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,
  preselect = TRUE,
  trunc_lvl = Inf,
  tree_crit = "tau",
  threshold = 0,
  keep_data = FALSE,
  show_trace = FALSE,
  cores = 1
)
```

**Arguments**

<code>data</code>	a matrix or data.frame (copula data should have approximately uniform margins).
<code>p</code>	the Markov order.
<code>var_types</code>	variable types; discrete variables not (yet) allowed.
<code>family_set</code>	a character vector of families; see <code>rvinecopulib::bicop()</code> for additional options.
<code>cs_structure</code>	the cross-sectional vine structure (see <code>rvinecopulib::rvine_structure()</code> ); <code>cs_structure = NA</code> performs automatic structure selection.
<code>out_vertices</code>	the out-vertex; if NA, the out-vertex is selected automatically if no structure is provided, and is equivalent to 1 if a structure is provided.
<code>in_vertices</code>	the in-vertex; if NA, the in-vertex is selected automatically if no structure is provided, and is equivalent to 1 if a structure is provided.
<code>type</code>	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.
<code>par_method</code>	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". For vinecop() there is the additional option "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	currently unsupported.
tree_crit	the criterion for tree selection, one of "tau", "rho", "hoeffd", or "mcor" for Kendall's $\tau$ , Spearman's $\rho$ , Hoeffding's $D$ , and maximum correlation, respectively.
threshold	for thresholded vine copulas; NA indicates that the threshold should be selected automatically by <code>rvinecopulib::mBICV()</code> .
keep_data	whether the data should be stored (necessary for using <code>fitted()</code> ).
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	<i>Custom S-vine models</i>
---------------	-----------------------------

---

### 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 'biscop_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	the out-vertex; if NA, the out-vertex is selected automatically if no structure is provided, and is equivalent to 1 if a structure is provided.
in_vertices	the in-vertex; if NA, the in-vertex is selected automatically if no structure is provided, and is equivalent to 1 if a structure is provided.
var_types	variable types; discrete variables not (yet) allowed.

### 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	<i>Expected hessian for S-vine copula models</i>
------------------	--

---

## 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 <a href="#">svinecop_dist</a> .
cores	number of cores to use; if larger than one, computations are done in parallel on cores batches .

## Value

Returns the observed Hessian matrix. Rows/columns correspond to 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\\_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	<i>Log-likelihood for S-vine copula models</i>
-----------------	--

---

**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 <a href="#">svinecop_dist</a> .
cores	number of cores to use; if larger than one, computations are done in parallel on cores batches .

**Value**

Returns the log-likelihood of the data for 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

<code>u</code>	the data; should have approximately uniform margins.
<code>model</code>	model inheriting from class <a href="#">svinecop_dist</a> .
<code>cores</code>	number of cores to use; if larger than one, computations are done in parallel on cores batches .

### Value

Returns a multivariate time series of pseudo-residuals

### 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 <a href="#">svinecop_dist</a> .
cores	number of cores to use; if larger than one, computations are done in parallel on cores batches .

### Value

A matrix containing the score vectors in its rows, where each row corresponds to one observation (row in u). 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	<i>Simulate from a S-vine copula model</i>
--------------	--

---

**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 <a href="#">svinecop_dist</a> ).
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 done 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](#).

## 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 of length `d` containing univariateML objects.  
`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

- `$margins`: list of marginal models from [univariateML](#),
- `$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 models*


---

### Description

Expected hessian of a parametric S-vine models

### Usage

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

### Arguments

x	the data.
model	S-vine model (inheriting from <a href="#">svine_dist</a> ).
cores	number of cores to use.

### Value

A returns a k-by-k matrix, where k is the total number of parameters in the model. 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)
Hi %*% I %*% t(Hi) / nrow(dat)

```

---

svine_loglik	<i>Log-likelihood for S-vine models</i>
--------------	---

---

**Description**

Log-likelihood for S-vine models

**Usage**

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

**Arguments**

x	the data.
model	model inheriting from class <a href="#">svine_dist</a> .
cores	number of cores to use; if larger than one, computations are done in parallel on cores batches .

**Value**

Returns the log-likelihood of the data for 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	<i>Pseudo-residuals of S-vine models</i>
------------------------	--

---

**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 done in parallel on cores batches .

**Value**

Returns a multivariate time series of pseudo-residuals

**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), ]))
```

---

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**

A returns a n-by-k matrix, where n = NROW(x) and k is the total number of parameters in the model. 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)
Hi %*% I %*% t(Hi) / nrow(dat)

```

---

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 <a href="#">svinecop_dist</a> ).
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 done 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

## \* datasets

returns, [2](#)

fitted(), [5](#)

returns, [2](#)

rvinecopulib, [5](#), [6](#)

rvinecopulib::bicop(), [4](#)

rvinecopulib::mBICV(), [5](#)

rvinecopulib::rvine\_structure(), [4](#)

svine, [2](#), [12](#)

svine\_bootstrap\_models, [3](#), [12](#), [13](#)

svine\_dist, [3](#), [13](#), [13](#), [14–16](#)

svine\_hessian, [14](#)

svine\_loglik, [3](#), [13](#), [15](#)

svine\_pseudo\_residuals, [15](#)

svine\_scores, [16](#)

svine\_sim, [3](#), [13](#), [17](#)

svinecop, [3](#)

svinecop\_dist, [6](#), [7–11](#), [17](#)

svinecop\_hessian, [6](#), [7](#), [10](#)

svinecop\_loglik, [6](#), [8](#)

svinecop\_pseudo\_residuals, [9](#)

svinecop\_scores, [6](#), [7](#), [10](#)

svinecop\_sim, [6](#), [11](#)

univariateML, [3](#), [13](#)