Package: portvine (via r-universe)

October 18, 2024

Title Vine Based (Un)Conditional Portfolio Risk Measure Estimation

Version 1.0.3.9000

Description Following Sommer (2022)
<https://mediatum.ub.tum.de/1658240> portfolio level risk
estimates (e.g. Value at Risk, Expected Shortfall) are
estimated by modeling each asset univariately by an ARMA-GARCH
model and then their cross dependence via a Vine Copula model
in a rolling window fashion. One can even condition on
variables/time series at certain quantile levels to stress test
the risk measure estimates.

License MIT + file LICENSE

URL https://github.com/EmanuelSommer/portvine,
https://emanuelsommer.github.io/portvine/

BugReports https://github.com/EmanuelSommer/portvine/issues

Depends R (>= 2.10)

Imports checkmate, data.table, dplyr, dtplyr, future.apply, methods,
ppcor, Rcpp (>= 0.12.12), rlang, rugarch, rvinecopulib, tidyr

Suggests covr, future, ggplot2, ggtext, knitr, patchwork, rmarkdown,
scales, testthat (>= 3.0.0)

LinkingTo BH, kde1d, Rcpp, RcppEigen, RcppThread, rvinecopulib, wdm

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

NeedsCompilation yes

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Collate 'RcppExports.R' 'default_garch_spec.R' 'S4_classes.R'
'datadoc_and_rcpp.R' 'dvine_ordering.R' 'risk_measures.R'
'rcondvinecop.R' 'estimate_dependence_and_risk.R'
'estimate_marginal_models.R' 'estimate_risk_roll.R' 'utils.R'

Repository https://emanuelsommer.r-universe.dev

RemoteUrl https://github.com/emanuelsommer/portvine

RemoteRef HEAD

RemoteSha f1e58e717c0f898d46f9e8ef786a9730cbc93644

Contents

`default_garch_spec` *Default specifications for ARMA-GARCH models*

Description

This function is used as the default for the univariate model fitting i.e. the marginal models and can be used to easily specify a different individual marginal model specification or default in `marginal_settings()`. The ARMA(`ar`,`ma`)-GARCH(`arch`,`garch`) is fitted with the distribution given by `dist` that specifies the conditional density used for the innovations.

Usage

```
default_garch_spec(ar = 1, ma = 1, arch = 1, garch = 1, dist = "sstd")
```

Arguments

`ar`	integer for the autoregressive order
`ma`	integer for the moving average order
`arch`	integer for the ARCH order
`garch`	integer for the GARCH order
`dist`	a single character value of the possible distributions allowed in `rugarch::ugarchspec`

Value

object of class rugarch::ugarchspec

See Also

marginal_settings()

Examples

```
# the default is then just using
default_garch_spec()
# to specify a ARMA(2,2)-GARCH(1,1) model with normal residual distribution
default_garch_spec(ar = 2, ma = 2, dist = "norm")
```

---

estimate_risk_roll *(Un-)conditional rolling risk estimation using vine copulas*

Description

As this is the main workhorse function with a lot going on under the hood it is advised to have a look at the vignettes or even better the package website as they provide a detailed hands on and theoretical documentation of what this function is doing and how it is intended to be used. For a short summarized explanation have a look at the Details section below.

Usage

```
estimate_risk_roll(
  data,
  weights = NULL,
  marginal_settings,
  vine_settings,
  alpha = 0.05,
  risk_measures = c("VaR", "ES_mean"),
  n_samples = 1000,
  cond_vars = NULL,
  cond_u = 0.05,
  n_mc_samples = 1000,
  trace = FALSE,
  cutoff_depth = NULL,
  prior_resid_strategy = FALSE
)
```

Arguments

data	Matrix, data.frame or other object coercible to a data.table storing the numeric asset returns in the named columns (at least 3). Moreover missing values must be imputed beforehand.

`weights` Corresponding named non-negative weights of the assets (conditioning variables must have weight 0). Default `NULL` gives equal weight of 1 to each non conditional asset. Alternatively one can use a matrix with as many rows as vine windows for changing weights. The matrix must have column names corresponding to the assets and conditional assets have to have weight 0.

`marginal_settings` `marginal_settings` S4 object containing the needed information for the ARMA-GARCH i.e. marginal models fitting. Note that the `marginal_settings` and `vine_settings` objects have to match as described further below.

`vine_settings` `vine_settings` S4 object containing the needed information for the vine copula model fitting. Note that the `marginal_settings` and `vine_settings` objects have to match as described further below.

`alpha` Numeric vector specifying the confidence levels in (0,1) at which the risk measures should be calculated.

`risk_measures` Character vector with valid choices for risk measures to estimate. Currently available are the Value at Risk `VaR` which is implemented in `est_var()` and 3 estimation methods of the Expected Shortfall `ES_mean`, `ES_median` and `ES_mc` all implemented in `est_es()` .

`n_samples` Positive count of samples to be used at the base of the risk measure estimation.

`cond_vars` Names of the variables to sample conditionally from (currently ≤ 2 variables).

`cond_u` Numeric vector specifying the corresponding quantiles in (0,1) of the conditional variable(s) conditioned on which the conditional risk measures should be calculated. Additionally always the conditioning values corresponding to the residual of one time unit prior are used as conditional variables (`cond_u` = 'prior_resid' in the risk measure output) if the flag`prior_resid` is set to TRUE, otherwise the conditioning values corresponding to the realized residual are used (`cond_u` = 'resid' in the risk measure output). The latter case corresponds to the default.

`n_mc_samples` Positive count of samples for the Monte Carlo integration if the risk measure `ES_mc` is used. (See `est_es()`)

`trace` If set to TRUE the algorithm will print a little information while running.

`cutoff_depth` Positive count that specifies the depth up to which the edges of the to be constructed D-vine copula are considered in the algorithm that determines the ordering for the D-vine fitting using partial correlations. The default `NULL` considers all edges and seems in most use cases reasonable. This argument is only relevant if D-vines are used.

`prior_resid_strategy` Logical flag that indicates whether as the additionally used conditioning values the prior day residual (if this flag is TRUE) or the realized residuals are used. The default are the realized residuals. Note that the resulting conditional risk measures use realized data so they are only for comparisons as they suffer from information leakage.

Details

Roughly speaking the function performs the following steps for the **unconditional risk measure estimation**:

- Fit for each asset marginal time series models i.e. ARMA-GARCH models in a rolling window fashion. The models as well as the rolling window size and training size are specified via the `marginal_settings` argument.
- Model the dependence between the assets with a vine copula model trained on the standardized residuals transformed to the copula scale via the probability integral transform. This is also performed in a rolling window fashion where one can use the same window size for the vine windows as used for the marginal ones or a smaller window size. This window size, the training size for the vine copula as well as the copula fitting arguments are specified via the `vine_settings` argument.
- Using the copula and the forecasted means and volatilities of the assets one simulates `n_samples` many forecasted portfolio level log returns for every time unit in every specified rolling window.
- Based on these samples one estimates portfolio level risk measures.

Additionally one can perform **conditional risk measure estimation** with up to two conditional log return series like market indices. Using this approach does not change the marginal models part but for the copula a D-vine with a special ordering i.e. the index or the indices are fixed as the rightmost leafs is fitted. One then simulates conditional forecasted portfolio log returns which then results in conditional risk measure estimates that can be particularly interesting in stress testing like situations. One conditions on a pre-specified quantile level (`cond_u`) of the conditioning assets (`cond_vars`) and for comparison one also conditions either on the behavior of the conditioning asset one time unit before (`prior_resid_strategy = TRUE`) or the realized behavior of the conditioning asset (`prior_resid_strategy = FALSE`).

Value

In the unconditional case an S4 object of class `portvine_roll` and in the conditional case its child class `cond_portvine_roll`. For details see `portvine_roll`.

Matching marginal and vine settings

First of all there must be at least 2 marginal windows. Thus `train_size + refit_size` slot in the `marginal_settings` class object must be smaller than the overall input data size. Moreover the `refit_size` of the marginal models must be dividable by the `refit_size` of the vine copula models e.g. possible combinations are 50 and 50, 50 and 25, 50 and 10. Furthermore the `train_size` of the vines must be smaller or equal to the `train_size` of the marginal models.

Parallel processing

This function uses the `future` framework for parallelization that allows maximum flexibility for the user while having safe speedups for example regarding random number generation. The default is of course the standard non parallel sequential evaluation. The user has to do nothing in order for this default to work. If the user wants to run the code in parallel there are many options from parallel on a single machine up to a high performance compute (HPC) cluster, all of this with just one setting switch i.e. by calling the function `future::plan()` with the respective argument before the function call. Common options are `future::plan("multisession")` which works on all major operating systems and uses all available cores to run the code in parallel local R sessions. To specify the number of workers use `future::plan("multisession", workers = 2)`. To go back to sequential processing and to shut down the parallel sessions use `future::plan("sequential")`.

For more information have a look at `future::plan()`. The two following loops are processed in parallel by default if a parallel `future::plan()` is set:

- The marginal model fitting i.e. all assets individually in parallel.
- The vine windows i.e. the risk estimates and the corresponding vine copula models are computed in parallel for each rolling vine window.

In addition the function allows for nested parallelization which has to be done with care. So in addition to the 2 loops above one can further run each computation for each time unit in the vine windows in parallel which might be especially interesting if the `n_samples` argument is large. Then the default parallelization has to be tweaked to not only parallelize the first level of parallelization which are the 2 loops above. This can be achieved e.g. via `future::plan(list(future::tweak(future::multisession, workers = 4), future::tweak(future::multisession, workers = 2)))`. This setting would run the 2 primary loops in 4 parallel R sessions and in addition each of the 4 primary parallel sessions would itself use 2 sessions within the nested parallel loop over the time units in the vine window. This results in a need for at least 2 times 4 so 8 threads on the hardware side. More details can be found in the extensive documentation of the `future` framework.

Author(s)

Emanuel Sommer

See Also

`portvine_roll`, `marginal_settings`, `vine_settings`, `est_var()`, `est_es()`

Examples

```
# For better illustrated examples have a look at the vignettes
# and/or the package website.

data("sample_returns_small")
ex_marg_settings <- marginal_settings(
  train_size = 900,
  refit_size = 50
)
ex_vine_settings <- vine_settings(
  train_size = 100,
  refit_size = 50,
  family_set = c("gaussian", "gumbel"),
  vine_type = "dvine"
)
# unconditionally
risk_roll <- estimate_risk_roll(
  sample_returns_small,
  weights = NULL,
  marginal_settings = ex_marg_settings,
  vine_settings = ex_vine_settings,
  alpha = c(0.01, 0.05),
  risk_measures = c("VaR", "ES_mean"),
  n_samples = 10,
  trace = FALSE
```

```
)
# conditional on one asset
risk_roll_cond <- estimate_risk_roll(
  sample_returns_small,
  weights = NULL,
  marginal_settings = ex_marg_settings,
  vine_settings = ex_vine_settings,
  alpha = c(0.01, 0.05),
  risk_measures = c("VaR", "ES_mean"),
  n_samples = 10,
  cond_vars = "GOOG",
  cond_u = c(0.05, 0.5),
  trace = FALSE,
  prior_resid_strategy = TRUE
)

# have a superficial look
risk_roll_cond
# a slightly more detailed look
summary(risk_roll_cond)

# actually use the results by extracting important fitted quantities
fitted_vines(risk_roll_cond)
fitted_marginals(risk_roll_cond)

# and of course most importantly the risk measure estimates
risk_estimates(
  risk_roll,
  risk_measures = "ES_mean",
  alpha = 0.05, exceeded = TRUE
)
risk_estimates(
  risk_roll_cond,
  risk_measures = "ES_mean",
  alpha = 0.05, exceeded = TRUE,
  cond_u = c("prior_resid", 0.5)
)
```

`est_es` *Estimate the Expected Shortfall (ES)*

Description

The Expected Shortfall at level α is defined as the expected value of the returns under the condition that the returns are smaller than the Value at Risk for the same α level. Note that an absolutely continuous distribution of the returns is assumed. The three estimation methods are:

- `mean` the mean of the samples that fall under the corresponding VaR.
- `median` the median of the samples that fall under the corresponding VaR.

- mc Calculation of the expected value using Monte Carlo integration over the α levels. One draws mc_samples Monte Carlo samples .

Usage

```
est_es(sample, alpha, method = c("mean", "median", "mc"), mc_samples = 100)
```

Arguments

sample	Numeric vector representing the sample upon which the Expected Shortfall is calculated.
alpha	Numeric vector with entries in (0,1) specifying the levels at which the ES is calculated.
method	Method of estimation one of mean, median, mc. For more information see the Description section.
mc_samples	Number of Monte Carlo samples used for the mc method.

Value

Numeric vector with Expected Shortfall estimates (same length as alpha).

See Also

est_var()

Examples

```
est_es(0:100, c(0.1, 0.2, 0.3))
```

---

est_var *Estimate the Value at Risk (VaR)*

Description

The VaR is defined as the empirical α level quantile of the empirical distribution based on a return sample.

Usage

```
est_var(sample, alpha)
```

Arguments

sample	Numeric vector representing the sample upon which the Value at Risk is calculated.
alpha	Numeric vector with entries in (0,1) specifying the levels at which the VaR is calculated.

Value

Numeric vector with VaR estimates (same length as `alpha`).

See Also

`est_es()`

Examples

```
est_var(0:100, c(0.1, 0.2, 0.3))
```

`fitted_marginals` *Accessor method for the fitted marginal models of* `(cond_)portvine_roll` *objects*

Description

Extract the marginal models that are ARMA-GARCH models which were fitted in a rolling window fashion using `rugarch::ugarchroll`. For the residual analysis of the models encompassed in such a `rugarch::ugarchroll` class object one can have a look at the utility function `roll_residuals()`.

Usage

```
fitted_marginals(roll, ...)

## S4 method for signature 'portvine_roll'
fitted_marginals(roll)
```

Arguments

`roll`	Object of class `portvine_roll` or a child class
`...`	Additional parameters for child class methods

Value

Named list with an entry for each asset containing a `rugarch::ugarchroll` class object that encompasses the marginal model fit.

See Also

`portvine_roll`, `roll_residuals()`

`fitted_vines` *Accessor method for the fitted vine copula models of* `(cond_)portvine_roll` *objects*

Description

Accessor method for the fitted vine copula models of `(cond_)portvine_roll` objects

Usage

```
fitted_vines(roll, ...)

## S4 method for signature 'portvine_roll'
fitted_vines(roll)
```

Arguments

`roll`	Object of class `portvine_roll` or a child class
`...`	Additional parameters for child class methods

Value

List of `rvinecopulib::vinecop` class objects each entry corresponds to one fitted vine copula model for the respective vine window.

See Also

`portvine_roll`

`marginal_settings-class` *S4 class for the marginal settings*

Description

Specify which marginal models (`individual_spec` & `default_specs`) are fitted and how often they are refit as well as how big the training data set is. Remember that the forecasting is done in a rolling window fashion and the arguments (train and refit size) will have to match with the arguments of the also to be specified `vine_settings`.

Usage

```
marginal_settings(
  train_size,
  refit_size,
  individual_spec = list(),
  default_spec = default_garch_spec()
)

## S4 method for signature 'marginal_settings'
show(object)
```

Arguments

train_size	equivalent to the slot definition below
refit_size	equivalent to the slot definition below
individual_spec	equivalent to the slot definition below
default_spec	equivalent to the slot definition below
object	An object of class marginal_settings

Details

For specifying the list for individual_spec or the argument default_spec the function default_garch_spec() might come in handy.

Value

Object of class marginal_settings

Functions

- marginal_settings(): Class constructor taking the arguments specified in the slots below

Slots

train_size Positive count specifying the training data size.

refit_size Positive count specifying size of the forecasting window.

individual_spec A named list. Specify ARMA-GARCH models for individual assets by naming the list entry as the asset and providing a rugarch::ugarchspec object.

default_spec rugarch::ugarchspec object specifying the default marginal model (used if the marginal model is not specified through individual_spec)

See Also

default_garch_spec(), vine_settings

Examples

```
# the most basic initialization
marginal_settings(train_size = 100, refit_size = 10)
# some individualism
marginal_settings(
  train_size = 100, refit_size = 10,
  individual_spec = list("GOOG" = default_garch_spec(ar = 3)),
  default_spec = default_garch_spec(dist = "norm")
)
```

---

portvine_roll-class *S4 output class for the function* estimate_risk_roll()

Description

The main output class for the function estimate_risk_roll()is portvine_roll but in the conditional case the child class cond_portvine_roll with some extra slots (below visible by the !C!) is returned.

Usage

```
## S4 method for signature 'portvine_roll'
show(object)

## S4 method for signature 'cond_portvine_roll'
show(object)

## S4 method for signature 'portvine_roll'
summary(object)

## S4 method for signature 'cond_portvine_roll'
summary(object)
```

Arguments

object	An object of class portvine_roll or cond_portvine_roll

Details

For easy access for the most important slots and some filtering functionality have a look at the accessor methods risk_estimates(), fitted_vines(), fitted_marginals().

Value

object of class portvine_roll

object of class cond_portvine_roll

Slots

`risk_estimates` data.table with the columns `risk_measure`, `risk_est`, `alpha`, `row_num`, `vine_window` and `realized` (here all samples also in the conditional case are used)

`fitted_marginals` named list with an entry for each asset containing a `rugarch::ugarchroll` class object that encompasses the marginal model fit.

`fitted_vines` list of `rvinecopulib::vinecop` class objects each entry corresponds to one vine window.

`marginal_settings` containing the specification used for the ARMA-GARCH fitting i.e. marginal models. Is of class `marginal_settings`.

`vine_settings` containing the specifications used for the vine fitting. Is of class `vine_settings`.

`risk_measures` a character vector displaying the estimated risk measures.

`alpha` numeric vector in (0,1) displaying the confidence levels used when estimating the risk measures.

`weights` the numeric positive weights of the assets. (Matrix with each row corresponding to one vine window) The weights of conditional variables are always 0.

`cond_estimation` logical value indicating whether the conditional estimation approach for the risk measures was used.

`n_samples` positive numeric count displaying how many return samples were used for the risk measure estimation.

`time_taken` numeric value displaying how many minutes the whole estimation process took.

`cond_risk_estimates` !C! data.table with the same columns as the `risk_estimate` slot has + the additional conditional columns with the respective conditioning value and the column character `cond_u` that indicates the used conditional quantile level or the conditional value corresponding to the residual one time unit prior with "prior_resid" or the realized residual with "resid".

`cond_vars` !C! character vector with the names of the variables that were used to sample conditionally from.

`cond_u` !C! a numeric vector specifying the corresponding quantiles in (0,1) of the conditional variable(s) conditioned on which the conditional risk measures were calculated.

See Also

`estimate_risk_roll()`, `risk_estimates()`, `fitted_vines()`, `fitted_marginals()`

`risk_estimates` *Accessor methods for the risk estimates of* `(cond_)portvine_roll` *objects*

Description

Accessor methods for the risk estimates of `(cond_)portvine_roll` objects

Usage

```
risk_estimates(
  roll,
  risk_measures = NULL,
  alpha = NULL,
  df = TRUE,
  exceeded = FALSE,
  ...
)

## S4 method for signature 'portvine_roll'
risk_estimates(
  roll,
  risk_measures = NULL,
  alpha = NULL,
  df = TRUE,
  exceeded = FALSE
)

## S4 method for signature 'cond_portvine_roll'
risk_estimates(
  roll,
  risk_measures = NULL,
  alpha = NULL,
  df = TRUE,
  exceeded = FALSE,
  cond = TRUE,
  cond_u = NULL
)
```

Arguments

roll	Object of class portvine_roll or a child class
risk_measures	Character vector of risk measures to filter for. Note that they must be fitted in the roll argument. The default will return all fitted risk measures
alpha	Numeric α levels of the estimated risk measures to filter for. Note that they must be fitted in the roll argument. The default will return all fitted α levels
df	Logical value if TRUE a data.frame is returned otherwise a data.table is returned.
exceeded	Logical value. If set to TRUE a column named exceeded will be appended that contains logical values telling whether the realized portfolio value exceeded the estimated risk.
...	Additional parameters for child class methods
cond	If set to TRUE returns the conditional risk estimates and otherwise returns the overall risk estimates.

cond_u — Numeric or character vector specifying the corresponding quantiles in (0,1) of the conditional variable(s) conditioned on which the conditional risk measures were calculated to filter for and/or the class "prior_resid"/"resid". Note that they must be fitted in the roll argument. The default will return all fitted risk measures.

Value

(Un-)filtered data.frame or data.table (see df argument) with at least the columns risk_measure, risk_est, alpha, row_num, vine_window and realized. exceeded column if the corresponding argument is set to TRUE. In the conditional case further columns are available (see: portvine_roll).

See Also

portvine_roll

---

roll_residuals *Extract fitted standardized residuals from a uGARCHroll object*

Description

The rugarch::ugarchroll class object encompasses fitting information about a number of models fitted in a rolling window fashion. This utility function gives an easy interface to extract the fitted residuals from one of these models. This can be especially helpful for assessing the model quality with a residual analysis.

Usage

```
roll_residuals(ugarchroll, roll_num = 1)
```

Arguments

ugarchroll — Object of class rugarch::ugarchroll.

roll_num — Count that specifies the fitted model to extract the residuals from.

Value

Numeric vector of the fitted standardized residuals.

sample_returns_small *A sample of log returns for 3 assets.*

Description

Data extracted from Yahoo Finance representing the daily log returns for Google, Apple, and Amazon stocks between 2014-01-13 and 2018-01-01 which results in exactly 1000 observations.

Usage

```
data(sample_returns_small)
```

Format

data.table with 3 columns and 1000 rows, columns `GOOG`, `AAPL`, `AMZN` contain the daily log return of the 3 stocks.

Source

Yahoo Finance

Examples

```
data(sample_returns_small)
head(sample_returns_small)
```

vine_settings-class *S4 class for the vine settings*

Description

Specify which vine copula models are fitted and how often they are refit as well as how big the training data set is. Remember that the estimation process is done in a rolling window fashion and the arguments (train and refit size) will have to match with the arguments of the also to be specified `marginal_settings`.

Usage

```
vine_settings(train_size, refit_size, family_set = "all", vine_type = "rvine")

## S4 method for signature 'vine_settings'
show(object)
```

Arguments

`train_size`	equivalent to the slot definition below
`refit_size`	equivalent to the slot definition below
`family_set`	equivalent to the slot definition below
`vine_type`	equivalent to the slot definition below
`object`	An object of class `vine_settings`

Value

Object of class `vine_settings`

Functions

- `vine_settings()`: Class constructor taking the arguments specified in the slots below

Slots

`train_size` Positive count specifying the training data size.

`refit_size` Positive count specifying for how many periods a vine is used

`family_set` Character vector specifying the family of copulas that are used. For possible choices see `rvinecopulib::bicop`. Note for conditional sampling just parametric copula families are possible so do not use the family arguments `all` and `tll`.

`vine_type` character value that specifies which vine class should be fitted. Possible choices right now are `rvine` (regular vine) and `dvine` (drawable vine).

See Also

`marginal_settings`

Examples

```
# the most basic initialization
vine_settings(100, 25)
# some individual note
vine_settings(
  train_size = 100, refit_size = 20,
  family_set = c("gumbel", "joe"),
  vine_type = "dvine"
)
```

Index