dynCopula Package Manual

2025-10-14

Contents

This document explains the dynamic copulas package **dynCopula**.

1 Introduction

Copula analysis provides a flexible framework for modeling complex dependencies, including non-linear relationships, tail dependence, and asymmetries. This flexibility makes it a powerful tool in finance, with applications in risk management (Embrechts et al., 2003; Giacomini et al., 2009),

asset allocation (Patton, 2004), and portfolio optimization. Accurately capturing the degree of dependence between assets is essential in these contexts.

The R package Copula offers tools for estimating both elliptical copulas (normal and Student-t) and Archimedean copulas (Clayton, Gumbel, and Frank). Yet, in many real-world situations, dependencies are not static. For instance, the relationships between international equity markets evolve over time, and static copulas may fail to capture these temporal dynamics, potentially leading to suboptimal investment decisions (Rockinger, 2001; Patton, 2006; Rodriguez, 2007).

The dynCopula package addresses this limitation by allowing the modeling of dynamic dependence between two time series. It provides estimation procedures for both elliptical and Archimedean copulas, offering a comprehensive toolkit for analyzing time-varying dependencies.

he package is structured around five core functions. The first, compute_innovations(), calculates the standardized innovations of the series under analysis. The second, fit_constant_copula(), estimates the parameters of a constant copula, which are subsequently used to initialize the estimation of the dynamic copula. The third function, negative_loglikelihood_dynamic_copula(), defines the negative log-likelihood of the selected copula. This function includes five sub-functions corresponding to the negative log-likelihood of each copula type: Normal, Student-t, Clayton, Gumbel, and Frank. The fourth function, optimize_time_varying_copula(), estimates the parameters governing the time-varying dynamics of the copula. Finally, the compute_dynamic_rho_tau() function allows the estimation of the time-varying copula parameter together with the associated Kendall's tau.

---

2 Main Functions of the Package

The main functions included in dynCopula are:

2.1 compute_innovations()

Computes standardized innovations based on selected GARCH models for two financial return series. Each series can have a different volatility model and can be estimated under various conditional error distributions.

Usage:

```
compute_innovations(series1, series2,
                    arma_order1, arma_order2,
                    garch_order1, garch_order2,
                    variance_model1, variance_model2,
                    submodel1, submodel2,
                    solver, distribution1, distribution2)
```

Arguments:

- series1, series2: numeric vectors of returns

- arma_order1, arma_order2: ARMA orders
- garch_order1, garch_order2: GARCH orders
- variance_model1, variance_model2: volatility models
- submodel1, submodel2: submodel types
- solver: optimization solver
- distribution1, distribution2: conditional error distributions

Value:

- A list with innov1 and innov2, vectors of standardized innovations.

Sort Example

```
# Simulated data

series1 = rt(1000, df= 50)
series2 = rt(1000, df=25)

# Compute innovations

innov <- compute_innovations(series1, series2,
                             arma_order1 = c(0,0),
                             arma_order2 = c(0,0),
                             garch_order1 = c(1,1),
                             garch_order2 = c(1,1),
                             variance_model1 = "fGARCH",
                             variance_model2 = "fGARCH",
                             submodel1 = "NAGARCH",
                             submodel2 = "NAGARCH",
                             solver = "solnp",
                             distribution1 = "std",
                             distribution2 = "std")

innov$innov1[1:5]  # First 5 innovations of series 1
innov$innov2[1:5]  # First 5 innovations of series 2
```

2.2 fit_constant_copula()

Estimates the parameters of a constant copula. These parameters are typically used as initial values for the estimation of a **dynamic copula**.

Usage:

```
fit_constant_copula(series1, series2,
                    copula_type = c("normal", "t", "clayton", "frank"),
                    param_init = NULL,
                    df_init = NULL)
```

Arguments:

- series1, series2: Numeric vectors of standardized innovations obtained from the compute_innovations() function.
- copula_type: Character string indicating the type of copula to estimate. Options are "normal", "t", "clayton", and "frank".
- param_init: Optional numeric value providing an initial guess for the copula parameter (e.g., correlation coefficient for the Normal copula). If NULL, a default initialization is used.
- df_init: Optional numeric value indicating the initial degrees of freedom when estimating a Student-t copula. Ignored for other copula types.

Value:

A list containing:

- parameters: Estimated copula parameters. For example:
  - For "normal": [rho]
  - For "t": [rho, nu]
  - For "clayton", "frank": [theta]
- logL: The corresponding log-likelihood value at the estimated parameters.

Short example:

```
parameters_constant_copula <- fit_constant_copula (
                              innov1,
                              innov2,
                              copula_type = "t",
                              param_init = NULL,
                              df_init = NULL)

# Extract estimated parameters
rho <- parameters_constant_copula$parameters[1]  # Correlation
nu  <- parameters_constant_copula$parameters[2]  # Degrees of freedom (if Student-t)
```

2.3 optimize_dynamic_copula()

Optimizes the **dynamic parameters** of a copula using the **negative log-likelihood function**. This function returns the optimized parameters for the dynamic copula model.

Usage:

```
optimize_dynamic_copula(theta0, rho, x, y,
                        copula_type = c("normal","t","clayton","gumbel","frank"),
                        nu = NULL)
```

Arguments:

- theta0: Initial guess for the dynamic parameters (theta1, theta2, theta3).
- rho: Initial copula parameter from the constant copula fit.
- x, y: Numeric vectors of standardized innovations (e.g., from compute_innovations()).
- copula_type: Type of copula to fit dynamically ("normal", "t", "clayton", "gumbel", "frank").
- nu: Degrees of freedom for Student-t copula (only if copula_type = "t").

Value:

A list containing:

- par: Optimized dynamic parameters (theta1, theta2, theta3).
- value: The value of the negative log-likelihood function at the optimum.

Short example

```
# Initial guess based on the constant copula parameter rho
aux_n  <- log((2 / (1 + as.numeric(rho))) - 1)
theta0 <- c(-aux_n, 0, 0)

# Optimize dynamic Student-t copula
dynamic_results <- optimize_dynamic_copula(
                   theta0,
                   rho,
                   x = innov1,
                   y = innov2,
                   copula_type = "t",
                   nu = nu)

# Extract optimized parameters and log-likelihood
theta_parameter <- dynamic_results$par
logL            <- dynamic_results$value
```

Special cases:

optimize_normal_copula(): Optimization routine for Normal copula

```
aux_n <- log((2 / (1 + as.numeric(rho))) - 1)
theta0 <- c(-aux_n, 0, 0)
```

optimize_student_copula(): Optimization routine for Student-t copula

```
aux_n <- log((2 / (1 + as.numeric(rho))) - 1)
theta0 <- c(-aux_n, 0, 0)
```

optimize_clayton_copula(): Optimization routine for Clayton copula

```
aux_n <- log(rho)
theta0 <- c(aux_n, 0, 0)
```

optimize_gumbel_copula(): Optimization routine for Gumbel copula

```
aux_n <- log(abs(rho-1))
theta0 <- c(aux_n, 0, 0)
```

optimize_frank_copula(): Optimization routine for Frank copula

```
aux_n <- rho
theta0 <- c(aux_n, 0, 0)
```

2.4 negative_loglikelihood_dynamic_copula()

Defines the **negative log-likelihood** of a dynamic copula, used internally for parameter optimization.
This function is embedded in `optimize_dynamic_copula()`, so users typically do **not** need to call it directly.

Usage:

```
negative_loglikelihood_dynamic_copula(theta, x, y,
                                      copula_type = c("normal",
                                                      "t","
                                                      clayton",
                                                      "gumbel",
                                                      "frank"),
                                      nu = NULL)
```

Arguments:

- theta: Vector of dynamic copula parameters (theta1, theta2, theta3).

- x, y: Numeric vectors of standardized innovations (from compute_innovations()).
- copula_type: Type of copula for which the negative log-likelihood is computed. Options: "normal", "t", "clayton", "gumbel", "frank".
- nu: Degrees of freedom for Student-t copula (only required if copula_type = "t").

Value:

- A numeric scalar representing the negative log-likelihood evaluated at the given parameters.

Short example

```
# Assume theta0 is an initial guess, and rho, innov1, innov2 exist
# This is just illustrative; do not execute in RMarkdown
neg_logL <- negative_loglikelihood_dynamic_copula(
             theta = theta0,
             x = innov1,
             y = innov2,
             copula_type = "t",
             nu = nu)
```

Special cases for each copula type:

negative_loglikelihood_normal_copula(): Negative log-likelihood for Normal copula

negative_loglikelihood_student_copula(): Negative log-likelihood for Student-t copula

negative_loglikelihood_clayton_copula(): Negative log-likelihood for Clayton copula

negative_loglikelihood_gumbel_copula(): Negative log-likelihood for Gumbel copula

negative_loglikelihood_frank_copula(): Negative log-likelihood for Frank copula

2.5 compute_dynamic_rho_tau()

Compute **dynamic copula parameters** and **Kendall's Tau** based on the optimized dynamic parameters.

Usage:

```
compute_dynamic_rho_tau(theta, rho, x, y,
                        copula_type = c("normal","t","clayton","gumbel","frank"),
                        nu = NULL)
```

Arguments:

- theta : Optimized dynamic copula parameters (theta1, theta2, theta3).
- rho : Constant copula parameter from fit_constant_copula().

- x, y : Numeric vectors of standardized innovations (from compute_innovations()).
- copula_type : Type of copula (“normal”, “t”, “clayton”, “gumbel”, “frank”).
- nu : Degrees of freedom for Student-t copula (only required if copula_type = “t”).

Value:

A list containing:

- param_t: Numeric vector of the time-varying copula parameter.
- tau : Numeric vector of the corresponding Kendall’s Tau values.

Short example:

```
# Compute dynamic copula parameter and Kendall's tau (do not execute)
results <- compute_dynamic_rho_tau(
             theta = theta_parameter,
             rho = rho,
             x = innov1,
             y = innov2,
             copula_type = "t",
             nu = nu)

time_varying_parameter <- results$param_t
time_varying_tau       <- results$tau
```

---

3 Example of Use

```
# ------------------------------------------------------------------------
# Example: Dynamic Normal Copula Estimation
# ------------------------------------------------------------------------

# 1. Simulate data
T <- 1000
series1  <- rnorm(T)
series2  <- rnorm(T)

# 2. Compute innovations
innovations <- compute_innovations(
  series1, series2,
  arma_order1 = c(1,0),
  arma_order2 = c(1,0),
```

```
  garch_order1 = c(1,1),
  garch_order2 = c(1,1),
  variance_model1 = "fGARCH",
  variance_model2 = "gjrGARCH",
  submodel1 = "NAGARCH",
  solver = "solnp",
  distribution1 = "norm",
  distribution2 = "norm"
)
innov1 <- innovations$innov1
innov2 <- innovations$innov2
stopifnot(length(innov1) == length(innov2))
pseudo_input <- cbind(innov1, innov2)

# 3. Estimate constant copula (normal)
parameters_constant_copula <- fit_constant_copula(
  pseudo_input[,1], pseudo_input[,2], copula_type = "normal"
)
rho   <- parameters_constant_copula$parameters
aux_n <- log((2 / (1 + as.numeric(rho))) - 1)
theta0 <- c(-aux_n, 0, 0)

# Show constant copula parameter
print(rho)
```

```
##        rho.1
## -0.006937694
```

```
# 4. Optimize dynamic copula
dynamic_results <- optimize_dynamic_copula(
  theta0, rho, x=innov1, y=innov2, copula_type="normal", nu = NULL
)
```

```
## final  value -0.028204
## converged
```

```
theta_parameters <- dynamic_results$par
logL             <- dynamic_results$value

# Show results
print(theta_parameters)
```

```
## [1] -0.0139215013  0.0004162683  0.0027751222
```

```r
print(logL)
```

```
## [1] -0.02820378
```

```r
# 5. Compute dynamic copula parameter and kendall'tau

library(knitr)
library(kableExtra)

results <- compute_dynamic_rho_tau(theta_parameters, rho,
                                   x = innov1, y = innov2,
                                   copula_type = "normal", nu = NULL)

time_varing_parameter <- results$param_t
time_varing_tau       <- results$tau


print(time_varing_parameter[1:5])
```

```
## [1] -0.006937694 -0.006896773 -0.007188895 -0.006987022 -0.007316949
```

```r
print(time_varing_tau[1:5])
```

```
## [1] -0.004416709 -0.004390657 -0.004576632 -0.004448113 -0.004658156
```

```r
# Resum table


stats <- data.frame(
  Estadistics       = c("Mean", "Mínimun", "Máximun"),
  Copula_Parameter  = c(mean(time_varing_parameter),
                        min(time_varing_parameter),
                        max(time_varing_parameter)),
  Kendall_Tau       = c(mean(time_varing_tau),
                        min(time_varing_tau),
                        max(time_varing_tau))
)


# Display table
kable(stats, caption = "Dynamic Copula Parameter and Kendall's Tau") %>%
  kable_styling(position = "center", full_width = FALSE) %>%
  column_spec(2:3, width = "3cm", border_left = TRUE, border_right = TRUE)
```

Table 1: Dynamic Copula Parameter and Kendall's Tau

Estadistics	Copula_Parameter	Kendall_Tau
Mean	-0.0069650	-0.0044341
Mínimun	-0.0080872	-0.0051485
Máximun	-0.0053251	-0.0033901

4 Package Structure

The dynCopula package has the following structure

- dynCopula/
  - R/ (Código fuente de las funciones)
    - compute_innovations.R
    - fit_constant_copula.R
    - negative_loglikelihood_copula.R
      - negative_loglikelihood_normal_copula.R
      - negative_loglikelihood_student_copula.R
      - negative_loglikelihood_clayton_copula.R
      - negative_loglikelihood_gumbel_copula.R
      - negative_loglikelihood_frank_copula.R
    - optimize_dynamic_copula.R
      - optimize_normal_copula.R
      - optimize_student_copula.R
      - optimize_clayton_copula.R
      - optimize_gumbel_copula.R
      - optimize_frank_copula.R
  - man/ (Documentación generada automáticamente)
  - DESCRIPTION (Metadatos del paquete)
  - NAMESPACE (Exportaciones de funciones)

Notas: - Each function is documented using roxygen2‘. - To update documentation:

```
roxygen2::roxygenise("C:/MisProyectos/R/dynCopula")
```

- To reinstall the package:

```
devtools::install("C:/MisProyectos/R/dynCopula")
library(dynCopula)
```

5 Estimations method

The traditional copula theory can be extended to the conditional case, allowing it to be used to analyze time-varying conditional dependence. The methodology applied in this package is based on Perez et al. (xxx), where the functional form of the copula described above remains fixed over the sample while its parameters vary according to an evolution equation. This approach has been followed by Hansen (1994) and Patton (2006). Patton proposes observation-driven copula models to specify the dynamics of the copula dependence parameter, in which the time-varying dependence parameter is a parametric function of transformations of lagged data and an autoregressive term. In this package, we assume that the functional form of the copulas remains fixed over the sample, whereas the parameters vary according to an evolution equation (Hansen, 1994). The parameters (ω, β, a) specify the copula dynamics. In the following lines, we display the dynamic equation for the copula parameter in the following cases: Normal, Student-t, Clayton, Gumbel, and Frank. For additional details about the estimation method, see Perez et al. (xxx).

5.1 Time varying Normal copula

The functional form of the time-varying normal copula parameter is given by:

$$\rho_t = \Lambda_1 \left\{ \omega + \beta \rho_{t-1} + \alpha \frac{1}{10} \sum_{j=1}^{10} \Phi^{-1}(u_{1,t-j})\, \Phi^{-1}(u_{2,t-j}) \right\}$$

where

$$\Lambda_1(x) = \frac{2}{1+e^{-x}} - 1$$

is the modified logistic transformation, designed to keep $\rho_t \in [-1, 1]$ at all times. The log-likelihood for the Normal copula estimation is given by:

$$\log L = \sum_{t=1}^{T} \frac{2\,\rho_t\, u_{1,t} u_{2,t} - u_{1,t}^2 - u_{2,t}^2}{2\,(1-\rho_t^2)}$$

5.2 Time varying Student-t copula

The functional form of the time-varying Student's t copula parameter is given by:

$$\rho_t = \Lambda_1 \left\{ \omega + \beta \rho_{t-1} + \alpha \frac{1}{10} \sum_{j=1}^{10} t_\nu^{-1}(u_{1,t-j})\, t_\nu^{-1}(u_{2,t-j}) \right\}$$

where

$$\Lambda_1(x) = \frac{2}{1+e^{-x}} - 1$$

is the modified logistic transformation, designed to keep $\rho_t \in [-1, 1]$ at all times.

The log-likelihood for the copula estimation (Bouyé et al., 2000) is given by:

$$\log L = \sum_{t=1}^{T}\Bigg[\Gamma \ln\left(\frac{\nu+2}{2}\right) + \Gamma \ln\left(\frac{\nu}{2}\right) - 2\Gamma \ln\left(\frac{\nu+1}{2}\right) - \frac{1}{2}\log(1-\rho_t^2) \\ - \frac{\nu+2}{2}\log\left(1 + \frac{u_{1,t}^2 + u_{2,t}^2 - 2\rho_t u_{1,t} u_{2,t}}{\nu(1-\rho_t^2)}\right) + \frac{\nu+1}{2}\log\Big(1 + \frac{u_{1,t}^2}{\nu}\Big) + \frac{\nu+1}{2}\log\Big(1 + \frac{u_{2,t}^2}{\nu}\Big)\Bigg]$$

The degree of freedom parameter remains fixed along the sample; it is set to the one estimated for the constant copula.

5.3 Time-varying Clayton Copula

The functional form of the time-varying Clayton copula parameter is given by:

$$\alpha_t = \Lambda_2\left\{\omega + \beta\alpha_{t-1} + \alpha\frac{1}{10}\sum_{j=1}^{10}|u_{1,t-j} - u_{2,t-j}|\right\}$$

where

$$\Lambda_2(x) = e^x$$

is the modified logistic transformation, designed to keep the Clayton parameter positive, $\alpha_t > 0$.

The log-likelihood for the Clayton copula estimation (Joe, 1997) is given by:

$$\log L = \sum_{t=1}^{T}\left[\log(1+\alpha_t) - (1+\alpha_t)\log(|u_{1,t}u_{2,t}|) - \frac{2}{1+\alpha_t}\log\Big(u_{1,t}^{-\alpha_t} + u_{2,t}^{-\alpha_t} - 1\Big)\right]$$

Time-varying Gumbel Copula

The functional form of the time-varying Gumbel copula parameter is given by:

$$\delta_t = \Lambda_3\left\{\omega + \beta\delta_{t-1} + \alpha\frac{1}{10}\sum_{j=1}^{10}|u_{1,t-j} - u_{2,t-j}|\right\}$$

where

$$\Lambda_3(x) = e^x + 1$$

is designed to maintain the Gumbel parameter $\delta_t > 1$.

The log-likelihood for the Gumbel copula estimation (Joe, 1997) is given by:

$$\log L = \sum_{t=1}^{T}\left[\log\left(\exp\Big(-\big((-\log u_{1,t})^{\delta_t} + (-\log u_{2,t})^{\delta_t}\big)^{1/\delta_t}\Big)\right) - \log(u_{1,t}) - \log(u_{2,t})\right]$$

5.4 Time-varying Frank Copula

The functional form of the time-varying Frank copula parameter is given by:

$$\theta_t = \omega + \beta\theta_{t-1} + \alpha\frac{1}{10}\sum_{j=1}^{10}|u_{1,t-j} - u_{2,t-j}|$$

where $\theta \in \mathbb{R}$.

The log-likelihood for the Frank copula estimation (Joe, 1997) is given by:

$$\log L = \sum_{t=1}^{T} 2\log\Big(1 - e^{-\theta_t} - (1 - e^{-\theta_t u_{1,t}})(1 - e^{-\theta_t u_{2,t}})\Big)$$

6 Next Steps

- Incorporate asymmetric copulas and additional families.
- Add validation and goodness-of-fit comparisons.
- Extend the manual with real data and empirical examples.

7 References

- Embrechts, P., Höing, A., & Juri, A. (2003). Using copulae to bound the value-at-risk for functions of dependent risks. Finance and Stochastics, 7, 145-167.
- Giacomini, E., Härdle, W., & Spokoiny, V. (2009). Inhomogeneous Dependence Modeling with Time-Varying Copulae. Journal of Business & Economic Statistics, 27(2), 224–234.
- Hansen, B.E. (1994), Autoregressive Conditional Density Estimation. International Economic Review 35, 705-30
- Joe, H. (1997). Multivariate Models and Dependence Concepts. London: Chapman & Hall.
- Jondeau, E., & Rockinger, M. (2006). The copula-garch model of conditional dependencies: An international stock market application. Journal of international money and finance, 25(5), 827-853.
- Patton, A. J. (2004). On the out-of-sample importance of skewness and asymmetric dependence for asset allocation. Journal of financial econometrics, 2(1), 130-168.
- Patton, A. J. (2006). Modelling asymmetric exchange rate dependence. International Economic Review, 2006, 47(2), 527-556.
- Rockinger, M., & Jondeau, E., (2001). Conditional Dependency of Financial Series: An Application of Copulas. Working Paper. CEPR and HEC.
- Rodriguez, J. C. (2007). Measuring Financial Contagion: A Copula Approach. Journal of Empirical Finance 14, 401-423