Package: BvCh (via r-universe)

October 21, 2026

Type Package

Title Bivariate Chen Distribution: Density, Survival, Hazard, Plots and Estimates

Version 0.1.0

Description Probability density, survival and hazard functions, random generation, graphical displays and maximum likelihood estimation for bivariate Chen type models. Three constructions are covered: the Bivariate Chen distribution with a singular component on the diagonal, the Bivariate Extended Chen distribution, and the Bivariate Inverse copula Chen distribution obtained through an inverse copula with Chen marginals. Univariate Chen density, distribution, quantile, survival, hazard and random generation routines are obtained from the 'reliaR' package.

License GPL-3

Encoding UTF-8

Depends R (>= 4.0.0)

Imports reliaR (>= 0.2), stats, graphics, grDevices

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/puneetstat999/BvCh

BugReports https://github.com/puneetstat999/BvCh/issues

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Mukul Bijalwan [aut] (ORCID iD: https://orcid.org/0009-0001-3040-6912), Puneet Kumar Gupta [aut, cre] (ORCID iD: https://orcid.org/0000-0002-3458-5655)

Maintainer Puneet Kumar Gupta <puneetstat999@gmail.com>

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

Date/Publication 2026-10-21 10:18:55 UTC

RemoteUrl https://github.com/cran/BvCh

RemoteRef HEAD

RemoteSha db4f9b561134f851889a09a526bbc2bb0be1d4e2

Contents

bec . . . 2
bvch . . . 4
bvch_components . . . 5
chen . . . 6
compare_models . . . 8
extendedchen . . . 9
fitbec . . . 10
fitbvch . . . 11
fitifgmchen . . . 12
hbec_grad . . . 13
ifgm . . . 14
ifgmchen . . . 16
ksfit_bvch . . . 17
loglik . . . 18
print.fitbvch . . . 19

Index **22**

`bec` *Bivariate Extended Chen distribution: density, survival, hazard and random generation*

Description

Probability density, survival function, hazard function and random generation for the Marshall-Olkin type Bivariate Extended Chen distribution $BEC(\alpha_1, \alpha_2, \alpha_3, \beta, \lambda)$.

Usage

```
dbec(x1, x2, alpha1, alpha2, alpha3, beta, lambda, log = FALSE)

sbec(x1, x2, alpha1, alpha2, alpha3, beta, lambda)

hbec(x1, x2, alpha1, alpha2, alpha3, beta, lambda)

rbec(n, alpha1, alpha2, alpha3, beta, lambda)
```

Arguments

`x1`, `x2`	numeric vectors of first and second component.
`alpha1`, `alpha2`, `alpha3`, `beta`, `lambda`	positive parameters.
`log`	logical; if `TRUE` the log density is returned.
`n`	number of observations.

Details

With $U_i \sim EC(\alpha_i, \beta, \lambda)$ independent and $X_1 = \min(U_1, U_3)$, $X_2 = \min(U_2, U_3)$, the joint survival function is

$$S(x_1, x_2) = \begin{cases} S_{EC}(x_1; \alpha_1, \beta, \lambda)\, S_{EC}(x_2; \alpha_2 + \alpha_3, \beta, \lambda) & x_2 > x_1 \\ S_{EC}(x_1; \alpha_1 + \alpha_3, \beta, \lambda)\, S_{EC}(x_2; \alpha_2, \beta, \lambda) & x_1 > x_2 \\ S_{EC}(x; \alpha_1 + \alpha_2 + \alpha_3, \beta, \lambda) & x_1 = x_2 = x, \end{cases}$$

where $S_{EC}(t; \alpha, \beta, \lambda) = [1 + \lambda(e^{t^\beta} - 1)]^{-\alpha}$ (see `sech`). The joint density is

$$f(x_1, x_2) = \begin{cases} f_1 & x_2 > x_1 \\ f_2 & x_1 > x_2 \\ f_3 & x_1 = x_2 = x, \end{cases}$$

with

$$f_1 = \alpha_1(\alpha_2 + \alpha_3)\lambda^2\beta^2 x_1^{\beta-1} x_2^{\beta-1} e^{x_1^\beta + x_2^\beta}[1 + \lambda(e^{x_1^\beta} - 1)]^{-(\alpha_1+1)}[1 + \lambda(e^{x_2^\beta} - 1)]^{-(\alpha_2+\alpha_3+1)},$$

$$f_2 = \alpha_2(\alpha_1 + \alpha_3)\lambda^2\beta^2 x_1^{\beta-1} x_2^{\beta-1} e^{x_1^\beta + x_2^\beta}[1 + \lambda(e^{x_1^\beta} - 1)]^{-(\alpha_1+\alpha_3+1)}[1 + \lambda(e^{x_2^\beta} - 1)]^{-(\alpha_2+1)},$$

$$f_3(x) = \alpha_3\beta\lambda x^{\beta-1} e^{x^\beta}[1 + \lambda(e^{x^\beta} - 1)]^{-(\alpha_1+\alpha_2+\alpha_3+1)}.$$

The three regions carry mass $\alpha_i/(\alpha_1 + \alpha_2 + \alpha_3)$, so the density integrates to one. The hazard function is f/S,

$$h(x_1, x_2) = \begin{cases} \dfrac{\alpha_1(\alpha_2 + \alpha_3)\lambda^2\beta^2 x_1^{\beta-1} x_2^{\beta-1} e^{x_1^\beta + x_2^\beta}}{[1 + \lambda(e^{x_1^\beta} - 1)][1 + \lambda(e^{x_2^\beta} - 1)]} & x_2 > x_1 \\ \dfrac{\alpha_2(\alpha_1 + \alpha_3)\lambda^2\beta^2 x_1^{\beta-1} x_2^{\beta-1} e^{x_1^\beta + x_2^\beta}}{[1 + \lambda(e^{x_1^\beta} - 1)][1 + \lambda(e^{x_2^\beta} - 1)]} & x_1 > x_2 \\ \dfrac{\alpha_3\beta\lambda x^{\beta-1} e^{x^\beta}}{1 + \lambda(e^{x^\beta} - 1)} & x_1 = x_2 = x. \end{cases}$$

The componentwise hazard gradients are available separately through `hbec1` and `hbec2`.

Value

A numeric vector, or for `rbec` a two column matrix with columns `x1` and `x2`.

See Also

`sbec`, `hbec`, `rbec`, `fitbec`, `sech`

Examples

```
dbec(0.5, 0.9, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8, lambda = 2)
sbec(0.5, 0.9, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8, lambda = 2)
hbec(0.5, 0.9, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8, lambda = 2)
set.seed(1)
rbec(5, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8, lambda = 2)
```

---

bvch — *Bivariate Chen distribution: density, survival, hazard and random generation*

Description

Probability density, survival function, hazard function and random generation for the Marshall-Olkin type Bivariate Chen distribution $BvCh(\alpha_1, \alpha_2, \alpha_3, \beta)$.

Usage

```
dbvch(z1, z2, alpha1, alpha2, alpha3, beta, log = FALSE)

sbvch(z1, z2, alpha1, alpha2, alpha3, beta)

hbvch(z1, z2, alpha1, alpha2, alpha3, beta)

rbvch(n, alpha1, alpha2, alpha3, beta)
```

Arguments

`z1`, `z2`	numeric vectors of first and second component.
`alpha1`, `alpha2`, `alpha3`, `beta`	positive parameters.
`log`	logical; if `TRUE` the log density is returned.
`n`	number of observations.

Details

Let $U_i \sim Chen(\alpha_i, \beta)$, $i = 1, 2, 3$, be independent and set $Z_1 = \min(U_1, U_3)$, $Z_2 = \min(U_2, U_3)$. Then the joint survival function is

$$S(z_1, z_2) = \exp\left\{\alpha_1\left(1 - e^{z_1^\beta}\right)\right\}\exp\left\{\alpha_2\left(1 - e^{z_2^\beta}\right)\right\}\exp\left\{\alpha_3\left(1 - e^{w^\beta}\right)\right\},$$

where $w = \max(z_1, z_2)$. Each factor is obtained from `schen`. The joint density is piecewise,

$$f(z_1, z_2) = \begin{cases} f_1(z_1, z_2) & z_1 < z_2 \\ f_2(z_1, z_2) & z_1 > z_2 \\ f_0(z) & z_1 = z_2 = z, \end{cases}$$

with

$$f_1 = \alpha_1(\alpha_2+\alpha_3)\beta^2 z_1^{\beta-1} z_2^{\beta-1} \exp\left\{z_1^\beta + z_2^\beta + \alpha_1(1-e^{z_1^\beta}) + (\alpha_2+\alpha_3)(1-e^{z_2^\beta})\right\},$$

$$f_2 = \alpha_2(\alpha_1+\alpha_3)\beta^2 z_1^{\beta-1} z_2^{\beta-1} \exp\left\{z_1^\beta + z_2^\beta + \alpha_2(1-e^{z_2^\beta}) + (\alpha_1+\alpha_3)(1-e^{z_1^\beta})\right\},$$

$$f_0(z) = \alpha_3\beta z^{\beta-1} \exp\left\{z^\beta + (\alpha_1+\alpha_2+\alpha_3)(1-e^{z^\beta})\right\}.$$

The mass carried by the three regions is respectively $\alpha_1/(\alpha_1+\alpha_2+\alpha_3)$, $\alpha_2/(\alpha_1+\alpha_2+\alpha_3)$ and $\alpha_3/(\alpha_1+\alpha_2+\alpha_3)$, so the density integrates to one. The hazard function is the ratio f/S, that is

$$h(z_1,z_2) = \begin{cases} \alpha_1(\alpha_2+\alpha_3)\beta^2 z_1^{\beta-1} z_2^{\beta-1} e^{z_1^\beta+z_2^\beta} & z_1 < z_2 \\ \alpha_2(\alpha_1+\alpha_3)\beta^2 z_1^{\beta-1} z_2^{\beta-1} e^{z_1^\beta+z_2^\beta} & z_1 > z_2 \\ \alpha_3\beta z^{\beta-1} e^{z^\beta} & z_1 = z_2 = z. \end{cases}$$

The model is a mixture (Proposition 1) of an absolutely continuous part of weight $(\alpha_1+\alpha_2)/(\alpha_1+\alpha_2+\alpha_3)$ and a singular part of weight $\alpha_3/(\alpha_1+\alpha_2+\alpha_3)$ supported on the diagonal; see dbvch_ac and dbvch_sing.

Value

A numeric vector, or for rbvch a two column matrix with columns z1 and z2.

See Also

sbvch, hbvch, rbvch, fitbvch, plot_surface_bvch

Examples

```
dbvch(0.5, 0.9, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8)
sbvch(0.5, 0.9, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8)
hbvch(0.5, 0.9, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8)
set.seed(1)
rbvch(5, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8)
```

---

bvch_components *Absolutely continuous and singular components of the BvCh density*

Description

Decomposition of the Bivariate Chen density into the two components of the mixture representation $f = \frac{\alpha_1+\alpha_2}{\alpha_1+\alpha_2+\alpha_3} f_a + \frac{\alpha_3}{\alpha_1+\alpha_2+\alpha_3} f_s$.

Usage

```
dbvch_ac(z1, z2, alpha1, alpha2, alpha3, beta)

dbvch_sing(z1, z2, alpha1, alpha2, alpha3, beta)
```

Arguments

z1, z2 numeric vectors of first and second component.

alpha1, alpha2, alpha3, beta positive parameters.

Details

dbvch_ac is the properly normalised absolutely continuous component of Proposition 1; it integrates to one over the plane, and dbvch_sing integrates to one along the diagonal. Consequently dbvch is recovered exactly as (alpha1 + alpha2)/(alpha1 + alpha2 + alpha3) * dbvch_ac(...) plus alpha3/(alpha1 + alpha2 + alpha3) * dbvch_sing(...).

Value

A numeric vector.

See Also

dbvch

Examples

```
a <- 1.5; b2 <- 2.5; c3 <- 0.5; bt <- 1.8
zz1 <- c(0.4, 0.9, 0.7); zz2 <- c(0.6, 0.4, 0.7)
ac <- dbvch_ac(zz1, zz2, a, b2, c3, bt)
sg <- dbvch_sing(zz1, zz2, a, b2, c3, bt)
k <- (a + b2 + c3)
mix <- (a + b2) / k * ac + c3 / k * sg
all.equal(mix, dbvch(zz1, zz2, a, b2, c3, bt))
```

---

chen *Univariate Chen distribution*

Description

Density, distribution function, quantile function, random generation, survival and hazard function for the univariate Chen distribution with shape parameter beta and scale parameter lambda.

Usage

```
dchen(x, beta, lambda, log = FALSE)

pchen(q, beta, lambda, lower.tail = TRUE, log.p = FALSE)

qchen(p, beta, lambda, lower.tail = TRUE, log.p = FALSE)

rchen(n, beta, lambda)
```

```
schen(x, beta, lambda)

hchen(x, beta, lambda)

hra.chen(x, beta, lambda)

crf.chen(x, t = 0, beta, lambda)
```

Arguments

`x`, `q`	vector of quantiles.
`beta`	shape parameter.
`lambda`	scale parameter.
`log`, `log.p`	logical; if `TRUE` probabilities are returned as `log(p)`.
`lower.tail`	logical; if `TRUE` (default) `P[X <= x]`, otherwise `P[X > x]`.
`p`	vector of probabilities.
`n`	number of observations.
`t`	age component for the conditional reliability function.

Details

These are the **reliaR** implementations, re-exported by **BvCh** so that the Chen building blocks used in the bivariate formulae are available without attaching a second package. The density is

$$f(x;\lambda,\beta) = \lambda\beta x^{\beta-1}\exp\left(x^{\beta}\right)\exp\left[\lambda\left\{1-\exp\left(x^{\beta}\right)\right\}\right],$$

and the survival function is

$$S(x;\lambda,\beta) = \exp\left[\lambda\left\{1-\exp(x^{\beta})\right\}\right].$$

Taking $\lambda = \alpha$ gives the $S_{Ch}(z;\alpha,\beta)$ that appears in the bivariate Chen formulae.

Value

`dchen` gives the density, `pchen` the distribution function, `qchen` the quantile function, `rchen` random deviates, `schen` the survival function and `hchen` the hazard function.

Random generation

`rchen` is *not* a direct call to `rchen`. In **reliaR** 0.2 that function does not agree with the package's own distribution, quantile and survival functions: for $\beta = 1.5$, $\lambda = 2$ its output has Kolmogorov-Smirnov distance about 0.38 from `pchen` ($p < 10^{-16}$), and the discrepancy grows with β (distance about 0.85 at $\beta = 2.5$); it happens to be correct only when $\beta = 1$. Since `qchen` is the verified analytic inverse of `pchen` (agreement to machine precision), `rchen` here draws `qchen(runif(n), beta, lambda)`, which keeps all six univariate Chen routines mutually consistent. Everything is still computed by **reliaR**.

See Also

Chen, Chensurvival

Examples

```
dchen(0.5, beta = 1.5, lambda = 2)
pchen(0.5, beta = 1.5, lambda = 2)
qchen(0.5, beta = 1.5, lambda = 2)
schen(0.5, beta = 1.5, lambda = 2)
hchen(0.5, beta = 1.5, lambda = 2)
```

---

compare_models *Model comparison table*

Description

Collects fitted models into a single table of $-\log L$, AIC and BIC.

Usage

```
compare_models(...)
```

Arguments

...	fitted model objects from fitbvch, fitbec or fitifgmchen.

Value

A data frame with columns Model, nobs, neglogL, AIC and BIC, sorted by increasing AIC so that the best model is on the first row.

Examples

```
set.seed(1)
a <- fitbvch(rbvch(200, 1.5, 2.5, 0.5, 1.8))
b <- fitbec(rbec(200, 1.5, 2.5, 0.5, 1.8, 2))
compare_models(a, b)
```

`extendedchen` *Univariate Extended Chen distribution*

Description

Density, distribution function, quantile function, random generation, survival and hazard function for the univariate Extended Chen distribution used as the latent building block of the Bivariate Extended Chen (BEC) model.

Usage

```
sech(x, alpha, beta, lambda)

hech(x, alpha, beta, lambda)

dech(x, alpha, beta, lambda, log = FALSE)

pech(q, alpha, beta, lambda, lower.tail = TRUE, log.p = FALSE)

qech(p, alpha, beta, lambda, lower.tail = TRUE, log.p = FALSE)

rech(n, alpha, beta, lambda)
```

Arguments

`x`, `q`	vector of quantiles.
`alpha`, `lambda`, `beta`	positive parameters.
`log`, `log.p`	logical; if `TRUE` probabilities are returned as `log(p)`.
`lower.tail`	logical; if `TRUE` (default) `P[X <= x]`, otherwise `P[X > x]`.
`p`	vector of probabilities.
`n`	number of observations.

Details

The Extended Chen survival function is

$$S_{EC}(x;\alpha,\beta,\lambda) = \left[1+\lambda\left(\exp(x^{\beta})-1\right)\right]^{-\alpha},$$

giving hazard

$$h_{EC}(x;\alpha,\beta,\lambda) = \frac{\alpha\lambda\beta x^{\beta-1}\exp(x^{\beta})}{1+\lambda\left(\exp(x^{\beta})-1\right)}$$

and density $f_{EC}(x) = S_{EC}(x)\,h_{EC}(x)$. The quantile function is available in closed form,

$$Q(p) = \left[\log\left\{1+\frac{p^{-1/\alpha}-1}{\lambda}\right\}\right]^{1/\beta}.$$

Unlike the univariate Chen distribution, which **reliaR** supplies, no Extended Chen implementation exists there, so it is provided here.

Value

dech gives the density, pech the distribution function, qech the quantile function, rech random deviates, sech the survival function and hech the hazard function.

Examples

```
sech(0.5, alpha = 1.5, beta = 1.5, lambda = 2)
hech(0.5, alpha = 1.5, beta = 1.5, lambda = 2)
qech(c(0.25, 0.5, 0.75), alpha = 1.5, beta = 1.5, lambda = 2)
```

---

fitbec — *Maximum likelihood estimation for the BEC model*

Description

Fits the Bivariate Extended Chen model by maximum likelihood.

Usage

```
fitbec(
  x1,
  x2 = NULL,
  start = NULL,
  singular = TRUE,
  method = "BFGS",
  control = list(),
  se = TRUE
)
```

Arguments

x1, x2	numeric vectors of observations, or a two column matrix.
start	optional starting values on the natural scale in the order alpha1, alpha2, alpha3, beta, lambda.
singular	logical; whether tied observations contribute the singular density.
method	optimisation method passed to optim.
control	optional control list for optim.
se	logical; whether standard errors should be computed.

Value

An object of class fitbec with the same components as fitbvch.

See Also

fitbvch, logLik_bec

Examples

```
set.seed(1)
d <- rbec(400, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8,
          lambda = 2)
fit <- fitbec(d)
fit
```

`fitbvch` *Maximum likelihood estimation for the BvCh model*

Description

Fits the Marshall-Olkin type Bivariate Chen model by maximum likelihood.

Usage

```
fitbvch(
  x1,
  x2 = NULL,
  start = NULL,
  singular = TRUE,
  method = "BFGS",
  control = list(),
  se = TRUE
)
```

Arguments

`x1`, `x2`	numeric vectors of observations, or a two column matrix.
`start`	optional numeric vector of starting values on the natural scale, in the order `alpha1`, `alpha2`, `alpha3`, `beta`.
`singular`	logical; whether tied observations contribute the singular density.
`method`	optimisation method passed to `optim`.
`control`	optional control list for `optim`.
`se`	logical; whether standard errors should be computed.

Details

Observations with `z1 == z2` fall on the singular component of the model. Whether they contribute the singular density f_0 or are discarded is controlled by `singular`; the counts are always reported so the choice stays visible. Standard errors come from the numerical Hessian of the log-likelihood and are on the original (untransformed) scale.

Value

An object of class `fitbvch`: a list with components `coefficients`, `vcov`, `se`, `loglik`, `aic`, `bic`, `nobs`, `counts`, `converged`, `message` and `call`.

See Also

`fitbec`, `fitifgmchen`, `logLik_bvch`

Examples

```
set.seed(1)
d <- rbvch(400, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8)
fit <- fitbvch(d)
fit
coef(fit)
```

---

fitifgmchen — *Maximum likelihood estimation for the IFGM-Chen model*

Description

Fits the Bivariate Inverse FGM Chen model by maximum likelihood.

Usage

```
fitifgmchen(
  x1,
  x2 = NULL,
  start = NULL,
  method = "BFGS",
  control = list(),
  se = TRUE
)
```

Arguments

`x1`, `x2`	numeric vectors of observations, or a two column matrix.
`start`	optional starting values on the natural scale in the order `theta1`, `beta1`, `theta2`, `beta2`, `gamma`, `omega`.
`method`	optimisation method passed to `optim`.
`control`	optional control list for `optim`.
`se`	logical; whether standard errors should be computed.

Details

The copula parameters gamma and omega are constrained to $[0, 1]$ and are estimated on the logit scale; the remaining parameters are estimated on the log scale. All reported values are on the natural scale.

Value

An object of class fitifgmchen with the same components as fitbvch.

Identifiability

The four marginal parameters are estimated very precisely, but gamma and omega are only weakly identified: with $n = 5000$ simulated from $(\gamma, \omega) = (0.6, 0.4)$ the marginal parameters have standard errors around 0.014 while gamma and omega have standard errors around 0.54 and 1.29. The log-likelihood surface is nearly flat in that direction, so individual values of gamma and omega should not be over-interpreted; logLik_ifgmchen is exported so the profile likelihood can be inspected and start used to explore it.

See Also

fitbvch, logLik_ifgmchen

Examples

```
set.seed(1)
d <- rifgmchen(400, theta1 = 1.2, beta1 = 1.3, theta2 = 1.5,
               beta2 = 1.1, gamma = 0.6, omega = 0.4)
fit <- fitifgmchen(d)
fit
```

---

hbec_grad — *Componentwise hazard gradients of the BEC model*

Description

The hazard gradients of the Bivariate Extended Chen model, that is the hazard of the i-th marginal conditional on the other component.

Usage

```
hbec1(x1, x2, alpha1, alpha2, alpha3, beta, lambda)

hbec2(x1, x2, alpha1, alpha2, alpha3, beta, lambda)
```

Arguments

`x1`, `x2` numeric vectors of first and second component.
`alpha1`, `alpha2`, `alpha3`, `beta`, `lambda`
positive parameters.

Details

$$h_{X_1}(x_1,x_2) = \begin{cases} \dfrac{\alpha_1\beta\lambda x_1^{\beta-1}e^{x_1^\beta}}{1+\lambda(e^{x_1^\beta}-1)} & x_2 > x_1 \\ \dfrac{(\alpha_1+\alpha_3)\beta\lambda x_1^{\beta-1}e^{x_1^\beta}}{1+\lambda(e^{x_1^\beta}-1)} & x_1 > x_2 \\ \dfrac{(\alpha_1+\alpha_2+\alpha_3)\beta\lambda x_1^{\beta-1}e^{x_1^\beta}}{1+\lambda(e^{x_1^\beta}-1)} & x_1 = x_2, \end{cases}$$

with h_{X_2} obtained by interchanging the roles of x_1, x_2 and α_1, α_2.

Value

A numeric vector.

See Also

hbec

Examples

```
hbec1(0.5, 0.9, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8, lambda = 2)
hbec2(0.5, 0.9, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8, lambda = 2)
```

---

ifgm *Inverse FGM copula*

Description

The Inverse FGM (IFGM) copula, its density, dependence measures and a random generator.

Usage

```
cifgm(u, v, gamma, omega)

cifgmdens(u, v, gamma, omega)

rifgm(n, gamma, omega)

tau_ifgm(gamma, omega)

rho_ifgm(gamma, omega)
```

Arguments

u, v	numeric vectors of copula arguments in $[0, 1]$.
gamma, omega	copula parameters, each in $[0, 1]$.
n	number of observations.

Details

For $0 \le \gamma, \omega \le 1$ the Inverse FGM copula is

$$C(u, v) = \frac{uv}{\sqrt{1 + \gamma(1-u)(1-v) + \omega uv(1-u)(1-v)}}.$$

Writing $D = 1 + \gamma(1-u)(1-v) + \omega uv(1-u)(1-v)$ and $P = uv$, its density has the closed form

$$c(u, v) = D^{-1/2} - \frac{vD_v + uD_u}{2D^{3/2}} + \frac{3PD_uD_v}{4D^{5/2}} - \frac{PD_{uv}}{2D^{3/2}},$$

with $D_u = -\gamma(1-v) + \omega(1-2u)v(1-v)$, $D_v = -\gamma(1-u) + \omega u(1-2v)(1-u)$ and $D_{uv} = \gamma + \omega(1-2u)(1-2v)$.

Kendall's tau and Spearman's rho are computed as

$$\tau = 4\int_0^1\int_0^1 C(u, v)c(u, v)\,du\,dv - 1, \qquad \rho = 12\int_0^1\int_0^1 uv\,c(u, v)\,du\,dv - 3.$$

Both are obtained by numerical quadrature and reproduce Monte Carlo estimates from `rifgm`.

Value

`cifgm` gives the copula, `cifgmdens` its density, `rifgm` a two column matrix of copula draws, and `tau_ifgm` and `rho_ifgm` scalar dependence measures.

Sign of the dependence

With the $+\gamma(1-u)(1-v) + \omega uv(1-u)(1-v)$ denominator given above, the copula lies *below* the independence copula uv ($C(0.5, 0.5) = 0.2306 < 0.25$ at $\gamma = 0.6, \omega = 0.4$), so τ and ρ are non-positive for $\gamma, \omega \ge 0$: numerically $\tau = -0.0699$ and $\rho = -0.1049$ at $\gamma = 0.6, \omega = 0.4$. This is the usual behaviour of an *inverse* FGM type copula. Both measures are exactly zero at $\gamma = \omega = 0$. Note that the summary document this package was written from quotes $\tau \approx 0.156 > 0$ for the same parameters, which is inconsistent with the copula formula it also quotes; the formula has been implemented as given, and `tau_ifgm`/`rho_ifgm` are provided so the dependence can be checked directly.

See Also

`difgmchen`, `hifgmchen`

Examples

```
cifgm(0.3, 0.7, gamma = 0.6, omega = 0.4)
cifgmdens(0.3, 0.7, gamma = 0.6, omega = 0.4)
tau_ifgm(0.6, 0.4)
rho_ifgm(0.6, 0.4)
set.seed(1)
head(rifgm(5, gamma = 0.6, omega = 0.4))
```

`ifgmchen` *Bivariate IFGM-Chen distribution*

Description

Reliability function, density, hazard function and random generation for the Bivariate Inverse FGM Chen model, obtained by combining the Inverse FGM copula with Chen marginals.

Usage

```
relifgmchen(x, y, theta1, beta1, theta2, beta2, gamma, omega)

difgmchen(x, y, theta1, beta1, theta2, beta2, gamma, omega, log = FALSE)

hifgmchen(x, y, theta1, beta1, theta2, beta2, gamma, omega)

rifgmchen(n, theta1, beta1, theta2, beta2, gamma, omega)
```

Arguments

`x`, `y`	numeric vectors of first and second component.
`theta1`, `beta1`, `theta2`, `beta2`	positive marginal parameters.
`gamma`, `omega`	copula parameters in $[0, 1]$.
`log`	logical; if `TRUE` the log density is returned.
`n`	number of observations.

Details

The marginals are Chen distributions $F_X(x) = 1 - e^{-\theta_1(e^{x^{\beta_1}}-1)}$ and $F_Y(y) = 1 - e^{-\theta_2(e^{y^{\beta_2}}-1)}$, which are exactly `reliaR::pchen(x, beta1, theta1)` and `reliaR::pchen(y, beta2, theta2)`. With C the Inverse FGM copula of `cifgm` and c its density, the joint reliability function and density are

$$R(x, y) = C(F_X(x), F_Y(y)) - F_X(x) - F_Y(y) + 1,$$
$$f(x, y) = c(F_X(x), F_Y(y))\, f_X(x)\, f_Y(y),$$

where $f_X = dchen(x, \beta_1, \theta_1)$ and $f_Y = dchen(y, \beta_2, \theta_2)$, where `dchen` is `dchen`. The hazard function is the ratio f/R.

Value

A numeric vector, or for `rifgmchen` a two column matrix with columns `x` and `y`.

Remarks

Unlike the Marshall-Olkin type models, this construction is absolutely continuous: it has no mass on the diagonal and therefore no singular component. The survival function is `1 - pchen(x, beta1, theta1) - pchen(y, beta2, theta2) + R(x, y)` and can be negative by cancellation in the far tail.

See Also

`cifgm`, `fitifgmchen`, `plot_reliability_ifgmchen`

Examples

```
difgmchen(0.5, 0.9, theta1 = 1.2, beta1 = 1.3, theta2 = 1.5,
          beta2 = 1.1, gamma = 0.6, omega = 0.4)
relifgmchen(0.5, 0.9, theta1 = 1.2, beta1 = 1.3, theta2 = 1.5,
            beta2 = 1.1, gamma = 0.6, omega = 0.4)
hifgmchen(0.5, 0.9, theta1 = 1.2, beta1 = 1.3, theta2 = 1.5,
          beta2 = 1.1, gamma = 0.6, omega = 0.4)
set.seed(1)
head(rifgmchen(5, theta1 = 1.2, beta1 = 1.3, theta2 = 1.5,
               beta2 = 1.1, gamma = 0.6, omega = 0.4))
```

`ksfit_bvch` *Kolmogorov-Smirnov goodness of fit for the BvCh marginals*

Description

Tests the fitted marginals of a Bivariate Chen fit against the data using `ks.chen`.

Usage

```
ksfit_bvch(fit, x1, x2 = NULL)
```

Arguments

`fit`	a fitted model from `fitbvch`.
`x1, x2`	numeric vectors of observations, or a two column matrix.

Details

The two components are compared with $Chen(\alpha_1 + \alpha_3, \beta)$ and $Chen(\alpha_2 + \alpha_3, \beta)$, since $Z_1 = \min(U_1, U_3)$ and $Z_2 = \min(U_2, U_3)$, and their minimum with $Chen(\alpha_1 + \alpha_2 + \alpha_3, \beta)$, which is the marginal of $\min(Z_1, Z_2) = \min(U_1, U_2, U_3)$. Each of these is a Chen distribution with the same shape β, because the minimum of independent Chen variables is Chen in the summed scale parameter.

Value

A data frame with one row per series (`Z1`, `Z2` and `min(Z1,Z2)`), giving the fitted `alpha` and `beta`, the Kolmogorov-Smirnov distance `D` and its p-value `p.value`, together with the sample mean and standard deviation.

Examples

```
set.seed(1)
d <- rbvch(200, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8)
fit <- fitbvch(d)
ksfit_bvch(fit, d)
```

---

loglik — *Log-likelihood functions of the bivariate Chen models*

Description

Log-likelihoods used by `fitbvch`, `fitbec` and `fitifgmchen`.

Usage

```
logLik_bvch(par, z1, z2, singular = TRUE)

logLik_bec(par, z1, z2, singular = TRUE)

logLik_ifgmchen(par, z1, z2)
```

Arguments

`par`	numeric vector of parameters on the log scale.
`z1`, `z2`	numeric vectors of observations.
`singular`	logical; if `TRUE` ties (`z1 == z2`) contribute the singular density. Use `FALSE` to drop tied observations.

Details

For the Marshall-Olkin type models the log-likelihood is the exact observed-data log-likelihood: the contribution of an observation is entirely determined by whether $z_1 < z_2$, $z_1 > z_2$ or $z_1 = z_2$, so the complete-data log-likelihood coincides with the observed-data one. Direct numerical maximisation is therefore used; it reaches the same fixed point as the EM algorithm.

Value

The sum of the log densities, or `-Inf` when any contribution is not finite.

Examples

```
set.seed(1)
d <- rbvch(50, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8)
logLik_bvch(log(c(1.5, 2.5, 0.5, 1.8)), d[, 1], d[, 2])
```

`print.fitbvch` *Print, summary and extractor methods for fitted bivariate Chen models*

Description

Methods for the objects returned by `fitbvch`, `fitbec` and `fitifgmchen`.

Usage

```
## S3 method for class 'fitbvch'
print(x, digits = 4, ...)

## S3 method for class 'fitbec'
print(x, digits = 4, ...)

## S3 method for class 'fitifgmchen'
print(x, digits = 4, ...)

## S3 method for class 'fitbvch'
summary(object, digits = 4, signif.stars = FALSE, ...)

## S3 method for class 'fitbec'
summary(object, digits = 4, signif.stars = FALSE, ...)

## S3 method for class 'fitifgmchen'
summary(object, digits = 4, signif.stars = FALSE, ...)

## S3 method for class 'fitbvch'
coef(object, ...)

## S3 method for class 'fitbec'
coef(object, ...)

## S3 method for class 'fitifgmchen'
coef(object, ...)

## S3 method for class 'fitbvch'
vcov(object, ...)

## S3 method for class 'fitbec'
vcov(object, ...)
```

```
## S3 method for class 'fitifgmchen'
vcov(object, ...)

## S3 method for class 'fitbvch'
nobs(object, ...)

## S3 method for class 'fitbec'
nobs(object, ...)

## S3 method for class 'fitifgmchen'
nobs(object, ...)

## S3 method for class 'fitbvch'
logLik(object, ...)

## S3 method for class 'fitbec'
logLik(object, ...)

## S3 method for class 'fitifgmchen'
logLik(object, ...)

## S3 method for class 'fitbvch'
print(x, digits = 4, ...)
## S3 method for class 'fitbec'
print(x, digits = 4, ...)
## S3 method for class 'fitifgmchen'
print(x, digits = 4, ...)
## S3 method for class 'fitbvch'
summary(object, digits = 4, signif.stars = FALSE, ...)
## S3 method for class 'fitbvch'
coef(object, ...)
## S3 method for class 'fitbec'
coef(object, ...)
## S3 method for class 'fitifgmchen'
coef(object, ...)
## S3 method for class 'fitbvch'
vcov(object, ...)
## S3 method for class 'fitbec'
vcov(object, ...)
## S3 method for class 'fitifgmchen'
vcov(object, ...)
## S3 method for class 'fitbvch'
nobs(object, ...)
## S3 method for class 'fitbec'
nobs(object, ...)
## S3 method for class 'fitifgmchen'
nobs(object, ...)
```

```
## S3 method for class 'fitbvch'
logLik(object, ...)
## S3 method for class 'fitbec'
logLik(object, ...)
## S3 method for class 'fitifgmchen'
logLik(object, ...)
```

Arguments

`x, object`	a fitted model object.
`digits`	number of significant digits to print.
`...`	further arguments, currently unused.
`signif.stars`	logical; whether to print significance stars.

Details

The fitted objects also carry `aic` and `bic` components, and `compare_models` tabulates them across models. Because `aic` and `bic` are not S3 generics that a package can register methods for, no `aic` or `bic` methods are provided.

Value

The printing methods return their argument invisibly; `coef`, `vcov`, `nobs` and `logLik` return the corresponding component of the fit.

Examples

```
set.seed(1)
d <- rbvch(300, alpha1 = 1.5, alpha2 = 2.5, alpha3 = 0.5, beta = 1.8)
fit <- fitbvch(d)
print(fit)
summary(fit)
coef(fit)
vcov(fit)
nobs(fit)
```

Index

bec, 2
bvch, 4
bvch_components, 5

Chen, *8*
chen, 6
Chensurvival, *8*
cifgm, *16, 17*
cifgm *(*ifgm*)*, 14
cifgmdens *(*ifgm*)*, 14
coef.fitbec *(*print.fitbvch*)*, 19
coef.fitbvch *(*print.fitbvch*)*, 19
coef.fitifgmchen *(*print.fitbvch*)*, 19
compare_models, 8, *21*
crf.chen *(*chen*)*, 6

dbec *(*bec*)*, 2
dbvch, *6*
dbvch *(*bvch*)*, 4
dbvch_ac, *5*
dbvch_ac *(*bvch_components*)*, 5
dbvch_sing, *5*
dbvch_sing *(*bvch_components*)*, 5
dchen, *16*
dchen *(*chen*)*, 6
dech *(*extendedchen*)*, 9
difgmchen, *15*
difgmchen *(*ifgmchen*)*, 16

extendedchen, 9

fitbec, *3*, *8*, 10, *12*, *18, 19*
fitbvch, *5*, *8*, *10*, 11, *13*, *17–19*
fitifgmchen, *8*, 12, *12*, *17–19*

hbec, *3*, *14*
hbec *(*bec*)*, 2
hbec1, *3*
hbec1 *(*hbec_grad*)*, 13
hbec2, *3*
hbec2 *(*hbec_grad*)*, 13
hbec_grad, 13
hbvch, *5*
hbvch *(*bvch*)*, 4
hchen *(*chen*)*, 6
hech *(*extendedchen*)*, 9
hifgmchen, *15*
hifgmchen *(*ifgmchen*)*, 16
hra.chen *(*chen*)*, 6

ifgm, 14
ifgmchen, 16

ks.chen, *17*
ksfit_bvch, 17

loglik, 18
logLik.fitbec *(*print.fitbvch*)*, 19
logLik.fitbvch *(*print.fitbvch*)*, 19
logLik.fitifgmchen *(*print.fitbvch*)*, 19
logLik_bec, *10*
logLik_bec *(*loglik*)*, 18
logLik_bvch, *12*
logLik_bvch *(*loglik*)*, 18
logLik_ifgmchen, *13*
logLik_ifgmchen *(*loglik*)*, 18

nobs.fitbec *(*print.fitbvch*)*, 19
nobs.fitbvch *(*print.fitbvch*)*, 19
nobs.fitifgmchen *(*print.fitbvch*)*, 19

optim, *10–12*

pchen *(*chen*)*, 6
pech *(*extendedchen*)*, 9
plot_reliability_ifgmchen, *17*
plot_surface_bvch, *5*
print.fitbec *(*print.fitbvch*)*, 19
print.fitbvch, 19
print.fitifgmchen *(*print.fitbvch*)*, 19

qchen *(*chen*)*, 6

qech (extendedchen), 9

rbec, *3*
rbec (bec), 2
rbvch, *5*
rbvch (bvch), 4
rchen, *7*
rchen (chen), 6
rech (extendedchen), 9
relifgmchen (ifgmchen), 16
rho_ifgm (ifgm), 14
rifgm (ifgm), 14
rifgmchen (ifgmchen), 16

sbec, *3*
sbec (bec), 2
sbvch, *5*
sbvch (bvch), 4
schen, *4*
schen (chen), 6
sech, *3*
sech (extendedchen), 9
summary.fitbec (print.fitbvch), 19
summary.fitbvch (print.fitbvch), 19
summary.fitifgmchen (print.fitbvch), 19

tau_ifgm (ifgm), 14

vcov.fitbec (print.fitbvch), 19
vcov.fitbvch (print.fitbvch), 19
vcov.fitifgmchen (print.fitbvch), 19