

# Package ‘dsfa’

July 19, 2023

**Title** Distributional Stochastic Frontier Analysis

**Version** 2.0.2

**Description** Framework to fit distributional stochastic frontier models. Casts the stochastic frontier model into the flexible framework of distributional regression or otherwise known as General Additive Models of Location, Scale and Shape (GAMLSS). Allows for linear, non-linear, random and spatial effects on all the parameters of the distribution of the output, e.g. effects on the production or cost function, heterogeneity of the noise and inefficiency. Available distributions are the normal-halfnormal and normal-exponential distribution. Estimation via the fast and reliable routines of the ‘mgcv’ package. For more details see <[doi:10.1016/j.csda.2023.107796](https://doi.org/10.1016/j.csda.2023.107796)>.

**Imports** mgcv, stats, Rdpack, Rcpp, RcppArmadillo, copula, gratia

**Suggests** plm, sfaR

**Depends** R (>= 3.5.0)

**NeedsCompilation** yes

**SystemRequirements** C++17

**License** MIT + file LICENSE

**LinkingTo** Rcpp, RcppArmadillo

**Encoding** UTF-8

**RdMacros** Rdpack

**LazyData** true

**RoxygenNote** 7.2.3

**Author** Rouven Schmidt [aut, cre]

**Maintainer** Rouven Schmidt <[rouven.schmidt@tu-clausthal.de](mailto:rouven.schmidt@tu-clausthal.de)>

**Repository** CRAN

**Date/Publication** 2023-07-19 21:40:16 UTC

## R topics documented:

BurkinaFarms . . . . .	2
BurkinaFarms_polys . . . . .	4

cdf2quantile . . . . .	4
chainrule . . . . .	5
comper . . . . .	6
comper_mv . . . . .	9
cop . . . . .	13
dcomper . . . . .	15
dcomper_mv . . . . .	17
dcop . . . . .	20
delta_bounds . . . . .	22
derivs_transform . . . . .	23
differencerule . . . . .	24
dnormexp . . . . .	25
dnormhnorm . . . . .	27
dsfa . . . . .	29
efficiency . . . . .	34
elasticity . . . . .	35
ind2joint . . . . .	37
list2derivs . . . . .	38
manuf . . . . .	39
mom2par . . . . .	40
par2mom . . . . .	41
productrule . . . . .	42
quotientrule . . . . .	43
sumrule . . . . .	44
transform . . . . .	45
trind . . . . .	46
trind_generator . . . . .	47
<b>Index</b>	<b>49</b>

---

BurkinaFarms

*BurkinaFarms*

---

## Description

Dataset of the Center for Environmental Economics and Policy in Africa (CEEPA), University of Pretoria. It contains information on the production of millet and sorghum in Burkina Faso.

## Usage

BurkinaFarms

## Format

BurkinaFarms is a data frame with 560 rows and 20 columns:

**hhcode** Respondent Households' identification number

**farmtype** Type of Farm Entity:

‘1‘ = small-scale

‘2‘ = medium-scale

‘3‘ = large-scale

**incfarm** Income of farmers

**educ1** education of household head in number of years

**adm1** Province/Region

**material** Light and heavy machinery as well as farm animal power multiplied with their price

**qharv\_millet** Quantity of millet harvested in kg

**lost\_millet** Quantity of millet lost due to disease and pests in kg

**seed\_millet** Amount of seeds used for the production of millet in kg

**land\_millet** Amount of land used for the production of millet in ha

**labour\_millet** Total estimated number of workdays for the production of millet. One day corresponds to 6-8 hours of work completed by one individual. Household labor and hired labor are accumulated

**fert\_millet** Amount of fertilizer used for the production of millet in kg

**pest\_millet** Amount of pesticides used for the production of millet in kg

**qharv\_sorghum** Quantity of sorghum harvested in kg

**lost\_sorghum** Quantity of sorghum lost due to disease and pests in kg

**seed\_sorghum** Amount of seeds used for the production of sorghum in kg

**land\_sorghum** Amount of land used for the production of sorghum in ha

**labour\_sorghum** Total estimated number of workdays for the production of sorghum. One day corresponds to 6-8 hours of work completed by one individual. Household labor and hired labor are accumulated

**fert\_sorghum** Amount of fertilizer used for the production of sorghum in kg

**pest\_sorghum** Amount of pesticides used for the production of sorghum in kg

## Details

This is a subset of the data which contains data only from Burkina Faso for the outputs sorghum and millet.

## Source

<[https://figshare.com/collections/An\\_agricultural\\_survey\\_for\\_more\\_than\\_9\\_500\\_African\\_households/1574094](https://figshare.com/collections/An_agricultural_survey_for_more_than_9_500_African_households/1574094)>

## References

Waha K, Zipf B, Kurukulasuriya P, Hassan RM (2016). "An agricultural survey for more than 9,500 African households." *Scientific data*, **3**(1), 1–8.

---

BurkinaFarms\_polys      *BurkinaFarms\_polys*

---

**Description**

Dataset of Burkina Faso - Subnational Administrative Boundaries

**Usage**

BurkinaFarms\_polys

**Format**

This file contains the polygons of the regions of Burkina Faso.

**Source**

<<https://data.humdata.org/dataset/burkina-faso-administrative-boundaries>>

---

cdf2quantile      *Inverse cumulative distribution function*

---

**Description**

Inverse cumulative distribution function

**Usage**

```
cdf2quantile(p, cdf, interval = c(-3, 3), ...)
```

**Arguments**

p	numeric vector of probabilities.
cdf	function, cumulative distribution function which to invert.
interval	numeric vector of length 2, determining the lower and upper bound of the uni-root interval
...	other arguments for the cdf, e.g. mu, sigma_v, sigma_u, s...

**Details**

Code is a clone from the package **gbutils**.

**Value**

Numeric vector of p evaluated in the inverse cdf.

**Examples**

```

q=5
cdf <- pnorm(q=q, mean=1, sd=2)
q_numeric <- cdf2quantile(p=cdf, cdf=pnorm, mean=1, sd=2)
all.equal(q,q_numeric)

```

chainrule

*Chainrule***Description**

Chainrule for derivs objects.

**Usage**

```
chainrule(f_list, tri, deriv_order)
```

**Arguments**

`f_list`, list of derivs objects of length  $M$ , e.g.  $list(f_1(\cdot), f_2(\cdot), \dots, f_M(\cdot))$   
`tri` list; created by the function `[trind_generator()]`.  
`deriv_order` integer; maximum order of derivative. Available are 0,2 and 4.

**Details**

Let  $f_m$  be a function defined in `[trind()]`, where  $m \in 1, \dots, M$ . Define  $h((x_{n1}, x_{n2}, \dots, x_{nK})) = f_1(\cdot) \circ f_2(\cdot) \dots \circ f_M(x_{n1}, x_{n2}, \dots, x_{nK})$ . In order to get the derivatives of  $h(\cdot)$  w.r.t all parameters  $x_{nk}$ , the chainrule is applied. For more details see `[trind()]` and `[trind_generator()]`.

**Value**

Returns an object of class `derivs` for the function  $h(\cdot)$ .

**See Also**

Other derivs: `derivs_transform()`, `differencerule()`, `ind2joint()`, `list2derivs()`, `productrule()`, `quotientrule()`, `sumrule()`, `trind_generator()`, `trind()`

**Examples**

```

A<-matrix(c(1:9)/10, ncol=1)
A_derivs<-list2derivs(list(A, A^0, A^2, A^3, A^4), deriv_order=4)
B_derivs<-transform(A, type="exp", par=0, deriv_order=4)
C_derivs<-transform(B_derivs, type="log", par=0, deriv_order=4)
chainrule(list(C_derivs, B_derivs), trind_generator(1), deriv_order=4) #equal to A_derivs

```

---

 comper

*comper*


---

### Description

The `comper` implements the composed-error distribution in which the  $\mu$ ,  $\sigma_V$  and  $\sigma_U$  can depend on additive predictors. Useable with `mgcv::gam`, the additive predictors are specified via a list of formulae.

### Usage

```
comper(
  link = list("identity", "logshift", "logshift"),
  s = -1,
  distr = "normhnorm",
  b = 0.01
)
```

### Arguments

<code>link</code>	three item list, specifying the link for the $\mu$ , $\sigma_V$ and $\sigma_U$ parameters. See details.
<code>s</code>	integer; $s = -1$ for production and $s = 1$ for cost function.
<code>distr</code>	string; determines the distribution: ‘normhnorm’, Normal-halfnormal distribution ‘normexp’, Normal-exponential distribution
<code>b</code>	positive parameter of the logshift link function.

### Details

Used with `gam()` to fit distributional stochastic frontier model. The function is called with a list containing three formulae:

1. The first formula specifies the response on the left hand side and the structure of the additive predictor for  $\mu$  parameter on the right hand side. Link function is "identity".
2. The second formula is one sided, specifying the additive predictor for the  $\sigma_V$  on the right hand side. Link function is "logshift", e.g.  $\log\{\sigma_V\} + b$ .
3. The third formula is one sided, specifying the additive predictor for the  $\sigma_U$  on the right hand side. Link function is "logshift", e.g.  $\log\{\sigma_U\} + b$ .

The fitted values and linear predictors for this family will be three column matrices. The first column is the  $\mu$ , the second column is the  $\sigma_V$ , the third column is  $\sigma_U$ . For more details of the distribution see `dcomper()`.

### Value

An object inheriting from class `general.family` of the `mgcv` package, which can be used in the `mgcv` and `dsfa` package.

## References

- Schmidt R, Kneib T (2023). “Multivariate distributional stochastic frontier models.” *Computational Statistics & Data Analysis*, 107796.
- Wood SN, Fasiolo M (2017). “A generalized Fellner-Schall method for smoothing parameter optimization with application to Tweedie location, scale and shape models.” *Biometrics*, **73**(4), 1071–1081.
- Aigner D, Lovell CK, Schmidt P (1977). “Formulation and estimation of stochastic frontier production function models.” *Journal of econometrics*, **6**(1), 21–37.
- Kumbhakar SC, Wang H, Horncastle AP (2015). *A practitioner’s guide to stochastic frontier analysis using Stata*. Cambridge University Press.
- Azzalini A (2013). *The skew-normal and related families*, volume 3. Cambridge University Press.
- Schmidt R, Kneib T (2020). “Analytic expressions for the Cumulative Distribution Function of the Composed Error Term in Stochastic Frontier Analysis with Truncated Normal and Exponential Inefficiencies.” *arXiv preprint arXiv:2006.03459*.

## Examples

```
### First example with simulated data
#Set seed, sample size and type of function
set.seed(1337)
N=500 #Sample size
s=-1 #Set to production function

#Generate covariates
x1<-runif(N,-1,1); x2<-runif(N,-1,1); x3<-runif(N,-1,1)
x4<-runif(N,-1,1); x5<-runif(N,-1,1)

#Set parameters of the distribution
mu=2+0.75*x1+0.4*x2+0.6*x2^2+6*log(x3+2)^(1/4) #production function parameter
sigma_v=exp(-1.5+0.75*x4) #noise parameter
sigma_u=exp(-1+sin(2*pi*x5)) #inefficiency parameter

#Simulate responses and create dataset
y<-rcomper(n=N, mu=mu, sigma_v=sigma_v, sigma_u=sigma_u, s=s, distr="normhnorm")
dat<-data.frame(y, x1, x2, x3, x4, x5)

#Write formulae for parameters
mu_formula<-y~x1+x2+I(x2^2)+s(x3, bs="ps")
sigma_v_formula<~1+x4
sigma_u_formula<~1+s(x5, bs="ps")

#Fit model
model<-dsfa(formula=list(mu_formula, sigma_v_formula, sigma_u_formula),
            data=dat, family=comper(s=s, distr="normhnorm"), optimizer = c("efs"))

#Model summary
summary(model)
```

```

#Smooth effects
#Effect of x3 on the predictor of the production function
plot(model, select=1) #Estimated function
lines(x3[order(x3)], 6*log(x3[order(x3)]+2)^(1/4)-
      mean(6*log(x3[order(x3)]+2)^(1/4)), col=2) #True effect

#Effect of x5 on the predictor of the inefficiency
plot(model, select=2) #Estimated function
lines(x5[order(x5)], -1+sin(2*pi*x5)[order(x5)]-
      mean(-1+sin(2*pi*x5)),col=2) #True effect

### Second example with real data

data("RiceFarms", package = "plm") #load data
RiceFarms[,c("goutput","size","seed", "totlabor", "urea")]<-
  log(RiceFarms[,c("goutput","size","seed", "totlabor", "urea")]) #log outputs and inputs
RiceFarms$id<-factor(RiceFarms$id) #id as factor

#Set to production function
s=-1

#Write formulae for parameters
mu_formula<-goutput ~ s(size, bs="ps") + s(seed, bs="ps") + #non-linear effects
  s(totlabor, bs="ps") + s(urea, bs="ps") + #non-linear effects
  varieties + #factor
  s(id, bs="re") #random effect
sigma_v_formula<--1
sigma_u_formula<--bimas

#Fit model with normhnorm dtribution
model<-dsfa(formula=list(mu_formula, sigma_v_formula, sigma_u_formula),
  data=RiceFarms, family=comper(s=-1, distr="normhnorm"), optimizer = "efs")

#Summary of model
summary(model)

#Plot smooths
plot(model)

### Third example with real data of cost function

data("electricity", package = "sfaR") #load data

#Log inputs and outputs as in Greene 1990 eq. 46
electricity$lcof<-log(electricity$cost/electricity$fprice)
electricity$lo<-log(electricity$output)
electricity$llf<-log(electricity$price/electricity$fprice)
electricity$lcf<-log(electricity$cprice/electricity$fprice)

#Set to cost function
s=1

```



```

#Write formulae for parameters
mu_formula<-lcof ~ s(lo, bs="ps") + s(llf, bs="ps") + s(lcf, bs="ps") #non-linear effects
sigma_v_formula<-~1
sigma_u_formula<-~s(lo, bs="ps") + s(lshare, bs="ps") + s(cshare, bs="ps")

#Fit model with normhnorm dtribution
model<-dsfa(formula=list(mu_formula, sigma_v_formula, sigma_u_formula),
             data=electricity, family=comper(s=s, distr="normhnorm"),
             optimizer = "efs")

#Summary of model
summary(model)

#Plot smooths
plot(model)

```

---

comper\_mv

*comper*


---

## Description

The `comper` implements the multivariate composed-error distribution in which the  $\mu_1$ ,  $\sigma_{V1}$ ,  $\sigma_{U2}$ ,  $\mu_2$ ,  $\sigma_{V2}$ ,  $\sigma_{U2}$  and  $\delta$  can depend on additive predictors. Useable with `mgcv::gam`, the additive predictors are specified via a list of formulae.

## Usage

```

comper_mv(
  link = list("identity", "logshift", "logshift", "identity", "logshift", "logshift",
             "glogit"),
  s = c(-1, -1),
  distr = c("normhnorm", "normhnorm", "normal"),
  rot = 0,
  b = 0.01
)

```

## Arguments

<code>link</code>	seven item list, specifying the links for $\mu_1$ , $\sigma_{V1}$ , $\sigma_{U2}$ , $\mu_2$ , $\sigma_{V2}$ , $\sigma_{U2}$ and $\delta$ . See details.
<code>s</code>	integer vector of length two; each element corresponds to one marginal.
<code>distr</code>	string vector of length three; the first two elements determine the distribution of the marginals. Available are: ‘normhnorm’, Normal-halfnormal distribution ‘normexp’, Normal-exponential distribution The last element determines the distribution of the copula:

	‘independent’, Independence copula
	‘normal’, Gaussian copula
	‘clayton’, Clayton copula
	‘gumbel’, Gumbel copula
	‘frank’, Frank copula
	‘joe’, Joe copula
	‘amh’, Ali-Mikhail-Haq copula
rot	integer determining the rotation for Archimedean copulas. Can be 90, 180 or 270.
b	positive parameter of the logshift link function.

### Details

Used with `gam()` to fit distributional stochastic frontier model. The function is called with a list containing three formulae:

1. The first formula specifies the response of marginal one on the left hand side and the structure of the additive predictor for  $\mu_1$  parameter on the right hand side. Link function is "identity".
2. The second formula is one sided, specifying the additive predictor for the  $\sigma_{V1}$  on the right hand side. Link function is "logshift", e.g.  $\log\{\sigma_{V1}\} + b$ .
3. The third formula is one sided, specifying the additive predictor for the  $\sigma_{U1}$  on the right hand side. Link function is "logshift", e.g.  $\log\{\sigma_{U1}\} + b$ .
4. The fourth formula specifies the response of marginal two on the left hand side and the structure of the additive predictor for  $\mu_2$  parameter on the right hand side. Link function is "identity".
5. The fifth formula is one sided, specifying the additive predictor for the  $\sigma_{V2}$  on the right hand side. Link function is "logshift", e.g.  $\log\{\sigma_{V2}\} + b$ .
6. The sixth formula is one sided, specifying the additive predictor for the  $\sigma_{U2}$  on the right hand side. Link function is "logshift", e.g.  $\log\{\sigma_{U2}\} + b$ .
7. The seventh formula is one sided, specifying the additive predictor for the  $\delta$  on the right hand side. Link function is "glogit".

The fitted values and linear predictors for this family will be seven column matrices. For more details of the distribution see `dcomper()`.

### Value

An object inheriting from class `general.family` of the `mgcv` package, which can be used in the `mgcv` and `dsfa` package.

### References

- Schmidt R, Kneib T (2023). “Multivariate distributional stochastic frontier models.” *Computational Statistics & Data Analysis*, 107796.
- Wood SN, Fasiolo M (2017). “A generalized Feller-Schall method for smoothing parameter optimization with application to Tweedie location, scale and shape models.” *Biometrics*, **73**(4), 1071–1081.

- Aigner D, Lovell CK, Schmidt P (1977). “Formulation and estimation of stochastic frontier production function models.” *Journal of econometrics*, 6(1), 21–37.
- Kumbhakar SC, Wang H, Horncastle AP (2015). *A practitioner’s guide to stochastic frontier analysis using Stata*. Cambridge University Press.
- Azzalini A (2013). *The skew-normal and related families*, volume 3. Cambridge University Press.
- Schmidt R, Kneib T (2020). “Analytic expressions for the Cumulative Distribution Function of the Composed Error Term in Stochastic Frontier Analysis with Truncated Normal and Exponential Inefficiencies.” *arXiv preprint arXiv:2006.03459*.

## Examples

```
#Set seed, sample size and type of function
set.seed(1337)
N=1000 #Sample size
s<-c(-1,-1) #Set to production function for margin 1 and set to cost function for margin 2

distr_cop="normal"
distr_marg1="normhnorm"
distr_marg2="normhnorm"

#Generate covariates
x1<-runif(N,-1,1); x2<-runif(N,-1,1); x3<-runif(N,-1,1)
x4<-runif(N,-1,1); x5<-runif(N,-1,1); x6<-runif(N,-1,1)
x7<-runif(N,-1,1)

mu1=6+2*x1+(-2/3)*x1^2 #production function parameter 1
sigma_v1=exp(-1.5+sin(2*pi*x2)) #noise parameter 1
sigma_u1=exp(-1) #inefficiency parameter 1
mu2=5*x4^2+4*log(x4+2)^(1/4) #cost function parameter 2
sigma_v2=exp(-1.5) #noise parameter 2
sigma_u2=exp(-1+sin(2*pi*x6)) #inefficiency parameter 2
delta=transform(x=matrix(1+2.5*cos(4*x7)),
  type="glogitinv",
  par=delta_bounds(distr_cop), deriv_order = 0)

#Simulate responses and create dataset
Y<-rcomper_mv(n=N, mu=cbind(mu1,mu2),
  sigma_v=cbind(sigma_v1, sigma_v2),
  sigma_u = cbind(sigma_u1, sigma_u2), s=s,
  delta=delta,
  distr = c(distr_marg1,distr_marg2,distr_cop))
dat<-data.frame(y1=Y[,1],y2=Y[,2], x1, x2, x3, x4, x5, x6, x7)

#Write formulae for parameters
mu_1_formula<-y1~s(x1,bs="ps")
sigma_v1_formula<-~s(x2,bs="ps")
sigma_u1_formula<-~1
mu_2_formula<-y2~s(x4,bs="ps")
sigma_v2_formula<-~1
```

```

sigma_u2_formula<-~s(x6,bs="ps")
delta_formula<-~s(x7,bs="ps")

#Fit model
model<-dsfa(formula=list(mu_1_formula,sigma_v1_formula,sigma_u1_formula,
                        mu_2_formula,sigma_v2_formula,sigma_u2_formula,
                        delta_formula), data=dat,
            family=comper_mv(s=s, distr=c(distr_marg1,distr_marg2,distr_cop)),
            optimizer="efs")

#Model summary
summary(model)

#Smooth effects
#Effect of x1 on the predictor of the production function of margin 1
plot(model, select=1) #Estimated function
lines(x1[order(x1)], 2*x1[order(x1)]+(-1/3)*x1[order(x1)]^2-
      mean(2*x1+(-1/3)*x1^2), col=2) #True effect

#Effect of x2 on the predictor of the noise of margin 1
plot(model, select=2) #Estimated function
lines(x2[order(x2)], -1.5+sin(2*pi*x2[order(x2)])-
      mean(-1.5+sin(2*pi*x2)),col=2) #True effect

#Effect of x4 on the predictor of the production function of margin 2
plot(model, select=3) #Estimated function
lines(x4[order(x4)], 3+5*x4[order(x4)]^2+4*log(x4[order(x4)]+2)^(1/4)-
      mean(3+5*x4^2+4*log(x4+2)^(1/4)), col=2) #True effect

#Effect of x6 on the predictor of the inefficiency of margin 2
plot(model, select=4) #Estimated function
lines(x6[order(x6)], -1+sin(2*pi*x6[order(x6)])-
      mean(-1+sin(2*pi*x6)),col=2) #True effect

#Effect of x7 on the predictor of the copula
plot(model, select=5) #Estimated function
lines(x7[order(x7)], 2.5*cos(4*x7[order(x7)])-
      mean(2.5*cos(4*x7)),col=2) #True effect

efficiency(model)
elasticity(model)

#' ### Second example with real data

data(BurkinaFarms)
data(BurkinaFarms_polys)

#Write formulae for parameters
mu_1_formula<-qharv_millet~s(land_millet, bs="ps")+s(labour_millet, bs="ps")+
  s(material, bs="ps")+s(fert_millet, bs="ps")+
  s(adm1, bs="mrf",xt=BurkinaFarms_polys)

sigma_v1_formula<-~1
sigma_u1_formula<-~farmtype+s(pest_millet, bs="ps")

```

```

mu_2_formula<-qharv_sorghum~s(land_sorghum, bs="ps")+s(labour_sorghum, bs="ps")+
  s(material, bs="ps")+s(fert_sorghum, bs="ps")+
  s(adm1, bs="mrf",xt=BurkinaFarms_polys)

sigma_v2_formula<~1
sigma_u2_formula<~farmtype+s(pest_sorghum, bs="ps")

delta_formula<~1

model<-dsfa(formula=list(mu_1_formula, sigma_v1_formula, sigma_u1_formula,
  mu_2_formula, sigma_v2_formula, sigma_u2_formula,
  delta_formula),
  data=BurkinaFarms,
  family=comper_mv(s=c(-1,-1),
  distr=c("normhnorm", "normhnorm", "normal")),
  optimizer="efs")

plot(model)

```

---

 cop

*cop*


---

## Description

The cop implements multiple copula distributions in which the parameter  $\delta$  can depend on additive predictors. Useable with `mgcv::gam`, the additive predictors are specified via a formula.

## Usage

```
cop(link = list("glogit"), W, distr = "normal", rot = 0)
```

## Arguments

link	formula, specifying the link for $\delta$ parameter. See details.
W	numeric matrix of pseudo observations. Must have two columns.
distr	string, defines the copula family: ‘independent’ = Independence copula ‘normal’ = Gaussian copula ‘clayton’ = Clayton copula ‘gumbel’ = Gumbel copula ‘frank’ = Frank copula ‘joe’ = Joe copula ‘amh’ = Ali-Mikhail-Haq copula
rot	integer determining the rotation for Archimedian copulas. Can be 90, 180 or 270.

## Details

Mostly internal function. Used with `gam` to fit copula model, which in turn is used for starting values. The function `gam` is from the `mgcv` package and is called with a formula. The formula specifies a dummy on the left hand side and the structure of the additive predictor for the  $\delta$  parameter on the right hand side. Link function is "generalized logit", where for each `distr` argument there are specific `min` and `max` arguments, which are the boundaries of the parameter space. Although the parameter space is larger in theory for some copulas, numeric under- and overflow limits the parameter space. The intervals for the parameter `delta` are provided by `[delta_bounds()]`. WARNING: Only the estimates of the coefficients are useful. The rest of the 'mgcv' object has no meaningful values, as `gam()` was more or less abused here.

## Value

An object inheriting from class `general.family` of the `mgcv` package, which can be used in the 'mgcv' and 'dsfa' package.

## References

- Schmidt R, Kneib T (2023). "Multivariate distributional stochastic frontier models." *Computational Statistics & Data Analysis*, 107796.
- Wood SN, Fasiolo M (2017). "A generalized Feller-Schall method for smoothing parameter optimization with application to Tweedie location, scale and shape models." *Biometrics*, 73(4), 1071–1081.
- Aigner D, Lovell CK, Schmidt P (1977). "Formulation and estimation of stochastic frontier production function models." *Journal of econometrics*, 6(1), 21–37.
- Kumbhakar SC, Wang H, Horncastle AP (2015). *A practitioner's guide to stochastic frontier analysis using Stata*. Cambridge University Press.
- Azzalini A (2013). *The skew-normal and related families*, volume 3. Cambridge University Press.
- Schmidt R, Kneib T (2020). "Analytic expressions for the Cumulative Distribution Function of the Composed Error Term in Stochastic Frontier Analysis with Truncated Normal and Exponential Inefficiencies." *arXiv preprint arXiv:2006.03459*.

## See Also

Other copula: `dcop()`, `delta_bounds()`

## Examples

```
#Set seed, sample size and type of copula
set.seed(1337)
N=500 #Sample size
cop="gumbel" #copula
rot=180 #rotation

#Generate covariates
x1<-runif(N,-1,1); x2<-runif(N,-1,1)
```

```

#Set parameters of the copula
eta<-matrix(1+2.5*x1+1.75*sin(pi*x2),nrow=N)
delta<-transform(x=eta, type="glogitinv", par=as.numeric(delta_bounds(cop)), deriv_order = 0)

#Simulate pseudo observations W and create dataset
dat<-as.data.frame(rcop(n=N, delta=delta, distr=cop, rot=rot))
dat$y<-1 #Add dummy response variable

#Write formulae for parameters
delta_formula<-y~x1+s(x2,bs="ps")

#Fit model
model<-mgcv::gam(delta_formula, data=dat,
                 family=cop(W=dat[,1:2],
                           distr=cop, rot=rot),
                 optimizer="efs")

#Smooth effects
#Effect of x2 on the predictor of delta
plot(model, select=1) #Estimated function
lines(x2[order(x2)], 1.75*sin(pi*x2[order(x2)])-
      mean(1.75*sin(pi*x2)), col=2) #True effect

```

---

dcomper

*Composed-Error distribution*


---

## Description

Probability density function, distribution, quantile function and random number generation for the composed-error distribution

## Usage

```

dcomper(
  x,
  mu = 0,
  sigma_v = 1,
  sigma_u = 1,
  s = -1,
  distr = "normhnorm",
  deriv_order = 0,
  tri = NULL,
  log.p = FALSE
)

pcomper(

```

```

    q,
    mu = 0,
    sigma_v = 1,
    sigma_u = 1,
    s = -1,
    distr = "normhnorm",
    deriv_order = 0,
    tri = NULL,
    log.p = FALSE
)

qcomper(
  p,
  mu = 0,
  sigma_v = 1,
  sigma_u = 1,
  s = -1,
  distr = "normhnorm",
  log.p = FALSE
)

rcomper(n, mu = 0, sigma_v = 1, sigma_u = 1, s = -1, distr = "normhnorm")

```

### Arguments

x	numeric vector of quantiles.
mu	numeric vector of $\mu$ .
sigma_v	numeric vector of $\sigma_V$ . Must be positive.
sigma_u	numeric vector of $\sigma_U$ . Must be positive.
s	integer; $s = -1$ for production and $s = 1$ for cost function.
distr	string; determines the distribution: 'normhnorm', Normal-halfnormal distribution 'normexp', Normal-exponential distribution
deriv_order	integer; maximum order of derivative. Available are 0,2 and 4.
tri	optional; index matrix for upper triangular, generated by <a href="#">trind_generator</a> .
log.p	logical; if TRUE, probabilities p are given as log(p).
q	numeric vector of quantiles.
p	numeric vector of probabilities.
n	positive integer; number of observations.

### Details

This is wrapper function for the normal-halfnormal and normal-exponential distribution. A random variable  $X$  follows a composed error distribution if  $X = V + s \cdot U$ , where  $V \sim N(\mu, \sigma_V^2)$  and  $U \sim HN(0, \sigma_U^2)$  or  $U \sim Exp(\sigma_U^2)$ . For more details see [dnormhnorm](#) and [dnormexp](#). Here,  $s = -1$  for production and  $s = 1$  for cost function.



**Value**

dcomper() gives the density, pcomper() give the distribution function, qcomper() gives the quantile function, and rcomper() generates random numbers, with given parameters. dcomper() and pcomper() returns a derivs object.

**Functions**

- pcomper(): distribution function for the composed-error distribution.
- qcomper(): quantile function for the composed-error distribution.
- rcomper(): random number generation for the composed-error distribution.

**References**

- Aigner D, Lovell CK, Schmidt P (1977). "Formulation and estimation of stochastic frontier production function models." *Journal of econometrics*, **6**(1), 21–37.
- Kumbhakar SC, Wang H, Horncastle AP (2015). *A practitioner's guide to stochastic frontier analysis using Stata*. Cambridge University Press.
- Schmidt R, Kneib T (2020). "Analytic expressions for the Cumulative Distribution Function of the Composed Error Term in Stochastic Frontier Analysis with Truncated Normal and Exponential Inefficiencies." *arXiv preprint arXiv:2006.03459*.
- Gradshteyn IS, Ryzhik IM (2014). *Table of integrals, series, and products*. Academic press.
- Azzalini A (2013). *The skew-normal and related families*, volume 3. Cambridge University Press.

**See Also**

Other distribution: [dcomper\\_mv\(\)](#), [dnormexp\(\)](#), [dnormhnorm\(\)](#)

**Examples**

```
pdf <- dcomper(x=5, mu=1, sigma_v=2, sigma_u=3, s=-1, distr="normhnorm")
cdf <- pcomper(q=5, mu=1, sigma_v=2, sigma_u=3, s=-1, distr="normhnorm")
q <- qcomper(p=seq(0.1, 0.9, by=0.1), mu=1, sigma_v=2, sigma_u=3, s=-1, distr="normhnorm")
r <- rcomper(n=10, mu=1, sigma_v=2, sigma_u=3, s=-1, distr="normhnorm")
```

---

dcomper\_mv

*Multivariate Composed-Error distribution*


---

**Description**

Probability density function, distribution, quantile function and random number generation for the multivariate composed-error distribution

**Usage**

```
dcomper_mv(
  x,
  mu = matrix(c(0, 0), ncol = 2),
  sigma_v = matrix(c(1, 1), ncol = 2),
  sigma_u = matrix(c(1, 1), ncol = 2),
  delta = matrix(0, nrow = 1),
  s = c(-1, -1),
  distr = c("normhnorm", "normhnorm", "normal"),
  rot = 0,
  deriv_order = 0,
  tri = NULL,
  log.p = FALSE
)
```

```
pcomper_mv(
  q,
  mu = matrix(c(0, 0), ncol = 2),
  sigma_v = matrix(c(1, 1), ncol = 2),
  sigma_u = matrix(c(1, 1), ncol = 2),
  delta = 0,
  s = c(-1, -1),
  distr = c("normhnorm", "normhnorm", "normal"),
  rot = 0,
  deriv_order = 0,
  tri = NULL,
  log.p = FALSE
)
```

```
rcomper_mv(
  n,
  mu = matrix(c(0, 0), ncol = 2),
  sigma_v = matrix(c(1, 1), ncol = 2),
  sigma_u = matrix(c(1, 1), ncol = 2),
  delta = matrix(0, nrow = 1),
  s = c(-1, -1),
  distr = c("normhnorm", "normhnorm", "normal"),
  rot = 0
)
```

**Arguments**

x	numeric matrix of quantiles. Must have two columns.
mu	numeric matrix of $\mu$ . Must have two columns.
sigma_v	numeric matrix of $\sigma_V$ . Must be positive and have two columns.
sigma_u	numeric matrix of $\sigma_U$ . Must be positive and have two columns.
delta	numeric vector of copula parameter $\delta$ .

s	integer vector of length two; each element corresponds to one marginal.
distr	string vector of length three; the first two elements determine the distribution of the marginals. Available are: ‘normhnorm’, Normal-halfnormal distribution ‘normexp’, Normal-exponential distribution The last element determines the distribution of the copula: ‘independent’, Independence copula ‘normal’, Gaussian copula ‘clayton’, Clayton copula ‘gumbel’, Gumbel copula ‘frank’, Frank copula ‘joe’, Joe copula ‘amh’, Ali-Mikhail-Haq copula
rot	integer determining the rotation for Archimedian copulas. Can be 90, 180 or 270.
deriv_order	integer; maximum order of derivative. Available are 0,2 and 4.
tri	optional; List of objects generated by [trind_generator()].
log.p	logical; if TRUE, probabilities p are given as log(p).
q	numeric matrix of probabilities.
n	positive integer; number of observations.

### Details

A bivariate random vector  $(X_1, X_2) = \mathbf{X}$  follows a multivariate composed-error distribution  $f_{X_1, X_2}(x_1, x_2)$ , which can be rewritten using Sklars’ theorem via a copula

$$f_{X_1, X_2}(y_1, y_2) = c(F_{X_1}(x_1), F_{X_2}(x_2), \delta) \cdot f_{X_1}(x_1) f_{X_2}(x_2) \quad ,$$

where  $c(\cdot)$  is the density of the copula and  $F_{X_m}(x_m), f_{X_m}(x_m)$  are the marginal cdfs and pdfs respectively for  $m \in \{1, 2\}$ .  $\delta$  is the copula parameter.

### Value

dcomper\_mv gives the density, pcomper\_mv give the distribution function, and rcomper\_mv generates random numbers, with given parameters. If the derivatives are calculated the output is a derivs object.

### Functions

- pcomper\_mv(): distribution function for the multivariate composed-error distribution.
- rcomper\_mv(): random number generation for the multivariate composed-error distribution.

### References

- Aigner D, Lovell CK, Schmidt P (1977). “Formulation and estimation of stochastic frontier production function models.” *Journal of econometrics*, **6**(1), 21–37.

- Kumbhakar SC, Wang H, Horncastle AP (2015). *A practitioner's guide to stochastic frontier analysis using Stata*. Cambridge University Press.
- Schmidt R, Kneib T (2020). "Analytic expressions for the Cumulative Distribution Function of the Composed Error Term in Stochastic Frontier Analysis with Truncated Normal and Exponential Inefficiencies." *arXiv preprint arXiv:2006.03459*.
- Gradshteyn IS, Ryzhik IM (2014). *Table of integrals, series, and products*. Academic press.
- Azzalini A (2013). *The skew-normal and related families*, volume 3. Cambridge University Press.

### See Also

Other distribution: [dcomper\(\)](#), [dnormexp\(\)](#), [dnormhnorm\(\)](#)

### Examples

```
pdf <- dcomper_mv(x=matrix(c(0,10),ncol=2), mu=matrix(c(1,2),ncol=2),
                 sigma_v=matrix(c(3,4),ncol=2), sigma_u=matrix(c(5,6),ncol=2),
                 delta=c(0.5), s=c(-1,-1), distr=c("normhnorm", "normhnorm", "normal"),
                 deriv=2 ,
                 tri=list(trind_generator(3),trind_generator(3),trind_generator(1),
                          trind_generator(6),trind_generator(7)),
                 log.p=TRUE)
cdf <- pcomper_mv(q=matrix(c(0,10),ncol=2), mu=matrix(c(1,2),ncol=2),
                 sigma_v=matrix(c(3,4),ncol=2), sigma_u=matrix(c(5,6),ncol=2),
                 delta=c(0.5), s=c(-1,-1), distr=c("normhnorm", "normhnorm", "normal"))
r <- rcomper_mv(n=10, mu=matrix(c(1,2),ncol=2),
               sigma_v=matrix(c(3,4),ncol=2), sigma_u=matrix(c(5,6),ncol=2),
               delta=c(0.5), s=c(-1,-1), distr=c("normhnorm", "normhnorm", "normal"))
```

---

dcop

*Copula function*

---

### Description

Probability density function, distribution and random number generation for copulas.

### Usage

```
dcop(
  W,
  delta,
  distr = "normal",
  rot = 0,
  deriv_order = 0,
  tri = NULL,
  log.p = FALSE
)
```

```
pcop(W, delta = 0, distr = "normal", rot = 0, log.p = FALSE)
```

```
rcop(n, delta = 0, distr = "normal", rot = 0)
```

### Arguments

W	numeric matrix of pseudo observations. Must have two columns.
delta	numeric vector of copula parameter $\delta$ .
distr	string, defines the copula family: ‘independent’ = Independence copula ‘normal’ = Gaussian copula ‘clayton’ = Clayton copula ‘gumbel’ = Gumbel copula ‘frank’ = Frank copula ‘joe’ = Joe copula ‘amh’ = Ali-Mikhail-Haq copula
rot,	integer determining the rotation for Archimedean copulas. Can be 90, 180 or 270.
deriv_order	integer; maximum order of derivative. Available are 0,2 and 4.
tri	optional; index matrix for upper triangular, generated by <a href="#">trind_generator</a> .
log.p	logical; if TRUE, probabilities p are given as log(p).
n	positive integer; number of observations.

### Details

A two-dimensional copula  $C(w_1, w_2, \delta)$  is a multivariate cumulative distribution function for which the marginal probability distribution of  $w_1$  and  $w_2$  are uniform on the interval  $[0, 1]$ . The parameter  $\delta$  specifies the copula.

The functions `pcop()` and `rcop()` are wrapper functions for the [pCopula](#) and [rCopula](#).

### Value

`dcop` gives the density, `pcop` gives the distribution function for a specified copula and `rcop` generates random numbers, with given delta. `dcop()` returns a `derivs` object. For more details see [trind](#) and [trind\\_generator](#).

### Functions

- `pcop()`: distribution function for copula.
- `rcop()`: random number generation for copula.

**References**

- Schepsmeier U, Stöber J (2014). “Derivatives and Fisher information of bivariate copulas.” *Statistical Papers*, **55**(2), 525–542.
- Hofert M, Kojadinovic I, Mächler M, Yan J (2018). *Elements of copula modeling with R*. Springer.

**See Also**

Other copula: [cop\(\)](#), [delta\\_bounds\(\)](#)

**Examples**

```
u=0.3; v=0.7; p=0.5
pdf <- dcop(W=cbind(u,v), delta=p, distr="normal")
cdf <- pcop(W=cbind(u,v), delta=p, distr="normal")
r <- rcop(n=100, delta=p, distr="normal")
```

---

delta\_bounds

*Bounds of Copula Parameter delta*

---

**Description**

Provides the minimum and maximum of the parameter space for  $\delta$

**Usage**

```
delta_bounds(distr)
```

**Arguments**

distr                    string, defines the copula family:  
                           ‘independent’ = Independence copula  
                           ‘normal’ = Gaussian copula  
                           ‘clayton’ = Clayton copula  
                           ‘gumbel’ = Gumbel copula  
                           ‘frank’ = Frank copula  
                           ‘joe’ = Joe copula  
                           ‘amh’ = Ali-Mikhail-Haq copula

**Details**

Although the parameter space is larger in theory for some copulas, numeric under- and overflow limits the parameter space. The parameter space of  $\delta$  is specified for each copula below:

- ‘independent’, min=0 and max=1

- ‘normal’, min=-1 and max=1
- ‘clayton’, min=1e-16 and max=28
- ‘gumbel’, min=1 and max=17
- ‘frank’, min=-35 and max=35
- ‘joe’, min=1e-16 and max=30
- ‘amh’, min=-1 and max=1

### Value

Returns numeric vector of length two with first argument being the minimum and the second argument being the maximum of the parameter space.

### See Also

Other copula: [cop\(\)](#), [dcop\(\)](#)

### Examples

```
delta_bounds("normal")
```

---

derivs_transform	<i>derivs_transform</i>
------------------	-------------------------

---

### Description

Transforms a derivs object via the specified function and applies the chainrule.

### Usage

```
derivs_transform(f, type, par, tri, deriv_order)
```

### Arguments

- |      |  |
|------|--|
| f    | derivs object.   |
| type | string, specifies the transformation function. Available are: <ol style="list-style-type: none"> <li>1. ‘identity’: <math>f(x) = x</math>.</li> <li>2. ‘exp’: <math>f(x) = \exp\{x\}</math>.</li> <li>3. ‘log’: <math>f(x) = \log\{x\}</math>.</li> <li>4. ‘glogit’: <math>f(x) = \log\{(-x + \text{min})/(x - \text{max})\}</math>, where <math>\text{par} = c(\text{min}, \text{max})</math>.</li> <li>5. ‘glogitinv’: <math>f(x) = \exp\{x\} \cdot (\text{max} + \text{min}) / (1 + \exp\{x\})</math>, where <math>\text{par} = c(\text{min}, \text{max})</math>.</li> <li>6. ‘inv’: <math>f(x) = \frac{1}{x}</math>.</li> <li>7. ‘pnorm’: <math>f(x) = \Phi(x)</math>.</li> <li>8. ‘qnorm’: <math>f(x) = \Phi^{-1}(x)</math>.</li> </ol> |

	9. 'mexp': $f(x) = -\exp\{x\}$ .
	10. 'zeta': $f(x) = \log\{2 \cdot \Phi(x)\}$ .
	11. 'constant': $f(x) = c$ .
	12. 'chainrule_utility': $f(x) = f'(x) = f''(x) = f'''(x) = f''''(x)$ .
	13. onemx: $1 - x$
par	numeric vector, additional parameters, e.g. min and max for glogit.
tri	list; created by the function [trind_generator()].
deriv_order	integer; maximum order of derivative. Available are 0,2 and 4.

### Details

Takes the derivs object `f` as an input for the function specified by `type` and evaluates it together with the derivatives utilizing the chainrule. For more details see [trind()] and [trind\_generator()].

### Value

Returns an object of class `derivs`

### See Also

Other derivs: [chainrule\(\)](#), [differencerule\(\)](#), [ind2joint\(\)](#), [list2derivs\(\)](#), [productrule\(\)](#), [quotientrule\(\)](#), [sumrule\(\)](#), [trind\\_generator\(\)](#), [trind\(\)](#)

### Examples

```
A<-matrix(c(1:9)/10, ncol=1)
A_mat<-list2derivs(list(A, A^0, A^2, A^3, A^4), deriv_order=4)
derivs_transform(f =derivs_transform(f = A, type="exp", par=0,
                                     tri=trind_generator(1), deriv_order=4),
                type="log", par=0, tri=trind_generator(1), deriv_order=4)
```

---

differencerule

*Differencerule*

---

### Description

Differencerule for derivs objects.

### Usage

```
differencerule(f_list, tri, deriv_order)
```

### Arguments

f_list	list of derivs objects of length $M$ , e.g. $list(f_1(\cdot), f_2(\cdot), \dots, f_M(\cdot))$
tri	list; created by the function [trind_generator()].
deriv_order	integer; maximum order of derivative. Available are 0,2 and 4.



**Details**

Let  $f_m$  be a function defined in `[trind()]`, where  $m \in 1, \dots, M$ . Define  $h((x_{n1}, x_{n2}, \dots, x_{nK})) = f_1(\cdot) - f_2(\cdot) \dots - f_M(x_{n1}, x_{n2}, \dots, x_{nK})$ . In order to get the derivatives of  $h(\cdot)$  w.r.t all parameters  $x_{nk}$ , the difference rule is applied. For more details see `[trind()]` and `[trind_generator()]`.

**Value**

Returns an object of class `derivs` for the function  $h(\cdot)$ .

**See Also**

Other derivs: `chainrule()`, `derivs_transform()`, `ind2joint()`, `list2derivs()`, `productrule()`, `quotientrule()`, `sumrule()`, `trind_generator()`, `trind()`

**Examples**

```
A<-matrix(c(1:9)/10, ncol=1)
A_derivs<-list2derivs(list(A, A^0, A^2, A^3, A^4), deriv_order=4)
differencerule(list(A_derivs, A_derivs), trind_generator(1), deriv_order=4) #equal to 0
```

---

dnormexp

*Normal-Exponential distribution*


---

**Description**

Probability density function, distribution, quantile function and random number generation for the normal-exponential distribution

**Usage**

```
dnormexp(
  x,
  mu = 0,
  sigma_v = 1,
  sigma_u = 1,
  s = -1,
  deriv_order = 0,
  tri = NULL,
  log.p = FALSE
)
```

```
pnormexp(
  q,
  mu = 0,
  sigma_v = 1,
  sigma_u = 1,
```

```

    s = -1,
    deriv_order = 0,
    tri = NULL,
    log.p = FALSE
  )

qnormexp(p, mu = 0, sigma_v = 1, sigma_u = 1, s = -1, log.p = FALSE)

rnormexp(n, mu = 0, sigma_v = 1, sigma_u = 1, s = -1)

```

### Arguments

x	numeric vector of quantiles.
mu	numeric vector of $\mu$ .
sigma_v	numeric vector of $\sigma_V$ . Must be positive.
sigma_u	numeric vector of $\sigma_U$ . Must be positive.
s	integer; $s = -1$ for production and $s = 1$ for cost function.
deriv_order	integer; maximum order of derivative. Available are 0,2 and 4.
tri	optional; index matrix for upper triangular, generated by <a href="#">trind_generator</a> .
log.p	logical; if TRUE, probabilities p are given as log(p).
q	numeric vector of quantiles.
p	numeric vector of probabilities.
n	positive integer; number of observations.

### Details

A random variable  $X$  follows a normal-exponential distribution if  $X = V + s \cdot U$ , where  $V \sim N(\mu, \sigma_V^2)$  and  $U \sim Exp(\sigma_U)$ . The density is given by

$$f_X(x) = \frac{\sigma_U}{2} \exp\{\sigma_U(s\mu) + \frac{1}{2}\sigma_U^2\sigma_V^2 - \sigma_U(sx)\} 2\Phi\left(\frac{1}{\sigma_V}(-s\mu) - \sigma_U\sigma_V + \frac{1}{\sigma_V}(sx)\right) \quad ,$$

where  $s = -1$  for production and  $s = 1$  for cost function. In the latter case the distribution is equivalent to the Exponentially modified Gaussian distribution. ’

### Value

dnormexp() gives the density, pnormexp() give the distribution function, qnormexp() gives the quantile function, and rnormexp() generates random numbers, with given parameters. dnormexp() and pnormexp() return a derivs object. For more details see [trind](#) and [trind\\_generator](#).

### Functions

- pnormexp(): distribution function for the normal-exponential distribution.
- qnormexp(): quantile function for the normal-exponential distribution.
- rnormexp(): random number generation for the normal-exponential distribution.

## References

- Aigner D, Lovell CK, Schmidt P (1977). “Formulation and estimation of stochastic frontier production function models.” *Journal of econometrics*, **6**(1), 21–37.
- Kumbhakar SC, Wang H, Horncastle AP (2015). *A practitioner’s guide to stochastic frontier analysis using Stata*. Cambridge University Press.
- Schmidt R, Kneib T (2020). “Analytic expressions for the Cumulative Distribution Function of the Composed Error Term in Stochastic Frontier Analysis with Truncated Normal and Exponential Inefficiencies.” *arXiv preprint arXiv:2006.03459*.
- Gradshteyn IS, Ryzhik IM (2014). *Table of integrals, series, and products*. Academic press.
- Azzalini A (2013). *The skew-normal and related families*, volume 3. Cambridge University Press.

## See Also

Other distribution: [dcomper\\_mv\(\)](#), [dcomper\(\)](#), [dnormhnorm\(\)](#)

## Examples

```
pdf <- dnormexp(x=5, mu=1, sigma_v=2, sigma_u=3, s=-1)
cdf <- pnormexp(q=5, mu=1, sigma_v=2, sigma_u=3, s=-1)
q <- qnormexp(p=seq(0.1, 0.9, by=0.1), mu=1, sigma_v=2, sigma_u=3, s=-1)
r <- rnormexp(n=10, mu=1, sigma_v=2, sigma_u=3, s=-1)
```

---

dnormhnorm

*Normal-halfnormal distribution*

---

## Description

Probability density function, distribution, quantile function and random number generation for the normal-halfnormal distribution

## Usage

```
dnormhnorm(
  x,
  mu = 0,
  sigma_v = 1,
  sigma_u = 1,
  s = -1,
  deriv_order = 0,
  tri = NULL,
  log.p = FALSE
)

pnormhnorm(
```

```

q,
mu = 0,
sigma_v = 1,
sigma_u = 1,
s = -1,
deriv_order = 0,
tri = NULL,
log.p = FALSE
)

qnormhnorm(p, mu = 0, sigma_v = 1, sigma_u = 1, s = -1)

rnormhnorm(n, mu = 0, sigma_v = 1, sigma_u = 1, s = -1)

```

### Arguments

x	numeric vector of quantiles.
mu	numeric vector of $\mu$ .
sigma_v	numeric vector of $\sigma_V$ . Must be positive.
sigma_u	numeric vector of $\sigma_U$ . Must be positive.
s	integer; $s = -1$ for production and $s = 1$ for cost function.
deriv_order	integer; maximum order of derivative. Available are 0,2 and 4.
tri	optional; index matrix for upper triangular, generated by <a href="#">trind_generator</a> .
log.p	logical; if TRUE, probabilities p are given as log(p).
q	numeric vector of quantiles.
p	numeric vector of probabilities.
n	positive integer; number of observations.

### Details

A random variable  $X$  follows a normal-halfnormal distribution if  $X = V + s \cdot U$ , where  $V \sim N(\mu, \sigma_V^2)$  and  $U \sim HN(\sigma_U^2)$ . The density is given by

$$f_X(x) = \frac{1}{\sqrt{\sigma_V^2 + \sigma_U^2}} \phi\left(\frac{x - \mu}{\sqrt{\sigma_V^2 + \sigma_U^2}}\right) \Phi\left(s \frac{\sigma_U}{\sigma_V} \frac{x - \mu}{\sqrt{\sigma_V^2 + \sigma_U^2}}\right),$$

where  $s = -1$  for production and  $s = 1$  for cost function.

### Value

dnormhnorm() gives the density, pnormhnorm() give the distribution function, qnormhnorm() gives the quantile function, and rnormhnorm() generates random numbers, with given parameters. dnormhnorm() and pnormhnorm() return a derivs object. For more details see [trind](#) and [trind\\_generator](#).

## Functions

- `pnormhnorm()`: distribution function for the normal-halfnormal distribution.
- `qnormhnorm()`: quantile function for the normal-halfnormal distribution.
- `rnormhnorm()`: random number generation for the normal-halfnormal distribution.

## References

- Aigner D, Lovell CK, Schmidt P (1977). “Formulation and estimation of stochastic frontier production function models.” *Journal of econometrics*, **6**(1), 21–37.
- Kumbhakar SC, Wang H, Horncastle AP (2015). *A practitioner’s guide to stochastic frontier analysis using Stata*. Cambridge University Press.
- Schmidt R, Kneib T (2020). “Analytic expressions for the Cumulative Distribution Function of the Composed Error Term in Stochastic Frontier Analysis with Truncated Normal and Exponential Inefficiencies.” *arXiv preprint arXiv:2006.03459*.
- Gradshteyn IS, Ryzhik IM (2014). *Table of integrals, series, and products*. Academic press.
- Azzalini A (2013). *The skew-normal and related families*, volume 3. Cambridge University Press.

## See Also

Other distribution: [dcomper\\_mv\(\)](#), [dcomper\(\)](#), [dnormexp\(\)](#)

## Examples

```
pdf <- dnormhnorm(x=5, mu=1, sigma_v=2, sigma_u=3, s=-1)
cdf <- pnormhnorm(q=5, mu=1, sigma_v=2, sigma_u=3, s=-1)
q <- qnormhnorm(p=seq(0.1, 0.9, by=0.1), mu=1, sigma_v=2, sigma_u=3, s=-1)
r <- rnormhnorm(n=10, mu=1, sigma_v=2, sigma_u=3, s=-1)
```

---

dsfa

*dsfa-package: Distributional Stochastic Frontier Analysis*

---

## Description

The **dsfa** package implements the specification, estimation and prediction of distributional stochastic frontier models via **mgcv**. The basic distributional stochastic frontier model is given by:

$$Y_n = \eta^\mu(\mathbf{x}_n^\mu) + V_n + s \cdot U_n$$

where  $n \in \{1, 2, \dots, N\}$ .  $V_n$  and  $U_n$  are the noise and (in)efficiency respectively.

- For  $s = -1$ ,  $\eta^\mu(\cdot)$  is the production function and  $\mathbf{x}_n^\mu$  are the log inputs. Alternatively, if  $s = 1$ ,  $\eta^\mu(\cdot)$  is the cost function and  $\mathbf{x}_n^\mu$  are the log cost. The vector  $\mathbf{x}_n^\mu$  may also contain other variables.

- The noise is represented as  $V_n \sim N(0, \sigma_{V_n}^2)$ , where  $\sigma_{V_n} = \exp(\eta^{\sigma_V}(\mathbf{x}_n^{\sigma_V}))$ . Here,  $\mathbf{x}_n^{\sigma_V}$  are the observed covariates which influence the parameter of the noise.
- The (in)efficiency can be represented in two ways.

- If  $U_n \sim HN(\sigma_{U_n}^2)$ , where  $\sigma_{U_n} = \exp(\eta^{\sigma_U}(\mathbf{x}_n^{\sigma_U}))$ . Here,  $\mathbf{x}_n^{\sigma_U}$  are the observed covariates which influence the parameter of the (in)efficiency. Consequently:

$$Y_n \sim \text{normhnorm}(\mu_n = \eta^\mu(\mathbf{x}_n^\mu), \sigma_{V_n} = \exp(\eta^{\sigma_V}(\mathbf{x}_n^{\sigma_V})), \sigma_{U_n} = \exp(\eta^{\sigma_U}(\mathbf{x}_n^{\sigma_U})), s = s)$$

. For more details see [dnormhnorm](#).

- If  $U_n \sim \text{Exp}(\sigma_{U_n})$ , where  $\sigma_{U_n} = \exp(\eta^{\sigma_U}(\mathbf{x}_n^{\sigma_U}))$ . Here,  $\mathbf{x}_n^{\sigma_U}$  are the observed covariates which influence the parameter of the (in)efficiency. Consequently:

$$Y_n \sim \text{normexp}(\mu_n = \eta^\mu(\mathbf{x}_n^\mu), \sigma_{V_n} = \exp(\eta^{\sigma_V}(\mathbf{x}_n^{\sigma_V})), \sigma_{U_n} = \exp(\eta^{\sigma_U}(\mathbf{x}_n^{\sigma_U})), s = s)$$

. For more details see [dnormexp](#).

Consequently,  $Y_n$  follows a composed-error distribution. For an overview see [dcomper](#).

Let  $\theta_n$  be a parameter of the distribution of  $Y_n$ , e.g.  $\theta_n \in \{\mu_n, \sigma_{U_n}, \sigma_{V_n}\}$ . Further, let  $g_\theta^{-1}(\cdot)$  be the monotonic response function, which links the additive predictor  $\eta(\mathbf{x}_n^\theta)$  to the parameter space for the parameter  $\theta_n$  via the additive model:

$$g_\theta^{-1}(\theta_n) = \eta(\mathbf{x}_n^\theta) = \beta_0^\theta + \sum_{j^\theta=1}^{J^\theta} h_{j^\theta}^\theta(x_{n,j^\theta}^\theta)$$

Thus, the additive predictor  $\eta(\mathbf{x}_n^\theta)$  is made up by the intercept  $\beta_0^\theta$  and  $J^\theta$  smooths terms. The **mgcv** packages provides a framework for fitting distributional regression models. For more information see [comper](#). The additive predictors can be defined via formulae in [gam](#). Within the formulae for the parameter  $\theta_n$ , the smooth function for the variable  $x_{n,j^\theta}^\theta$  can be specified via the function **s**, which is  $h_{j^\theta}^\theta(\cdot)$  in the notation above. The smooth functions may be:

- linear effects, may include polynomials or regression splines.
- non-linear effects, which can be modeled via penalized regression splines, e.g. [p.spline](#), [tprs](#).
- random effects, [random.effects](#).
- spatial effects, which can be modeled via [mrf](#).

An overview is provided at [smooth.terms](#). The functions [gam](#), [predict.gam](#) and [plot.gam](#), are alike to the basic S functions. A number of other functions such as [summary.gam](#), [residuals.gam](#) and [anova.gam](#) are also provided, for extracting information from a fitted [gamObject](#).

The main functions are:

- [comper](#) Object which can be used to fit a composed-error stochastic frontier model with the [mgcv](#) package.
- [comper\\_mv](#) Object which can be used to fit a multivariate composed-error stochastic frontier model with the [mgcv](#) package.
- [elasticity](#) Calculates and plots the elasticity of a smooth function.
- [efficiency](#) Calculates the expected technical (in)efficiency index  $E[U|\mathcal{E}]$  or  $E[\exp(-U)|\mathcal{E}]$ .

Further useful functions are:

- [dcomper](#) Probability density function, distribution, quantile function and random number generation for the composed-error distribution.
- [dcomper\\_mv](#) Probability density function, distribution, quantile function and random number generation for the multivariate composed-error distribution.
- [dcop](#) Probability density function, distribution and random number generation for copulas.

These are written in C++ for fast and accurate evaluation including derivatives. They may be helpful for other researchers, who want to avoid the tedious implementation. Additionally:

- [cop](#) Object which can be used to fit a copula with the **mgcv** package.

## Usage

```
dsfa(
  formula,
  family = comper(link = list("identity", "logshift", "logshift"), s = -1, distr =
    "normhnorm"),
  data = list(),
  optimizer = "efs",
  ...
)
```

## Arguments

formula	A list of formulas specifying the additive predictors. See <a href="#">formula.gam</a> and <a href="#">gam.models</a> for more details.
family	The family object specifies the (multivariate) composed-error distribution and link of the model. See <a href="#">comper</a> and <a href="#">comper_mv</a> for more details.
data	A data frame or list containing the model response variable and covariates required by the formula. By default the variables are taken from <code>environment(formula)</code> : typically the environment from which <code>dsfa</code> is called.
optimizer	An array specifying the numerical optimization method to use to optimize the smoothing parameter estimation criterion (given by method). "outer" for the more stable direct approach. "outer" can use several alternative optimizers, specified in the second element of optimizer: "newton" (default), "bfgs", "optim", "nlm" and "nlm.fd" (the latter is based entirely on finite differenced derivatives and is very slow). "efs" for the extended Fellner Schall method of Wood and Fasiolo (2017).
...	other parameters of <a href="#">gam</a>

## Details

This function is a wrapper for [gam](#).

## Value

Returns a [gam](#) object.

**Author(s)**

- Rouven Schmidt <rouven.schmidt@tu-clausthal.de>

**References**

- Schmidt R, Kneib T (2023). “Multivariate distributional stochastic frontier models.” *Computational Statistics & Data Analysis*, 107796.
- Wood SN, Fasiolo M (2017). “A generalized Fellner-Schall method for smoothing parameter optimization with application to Tweedie location, scale and shape models.” *Biometrics*, 73(4), 1071–1081.
- Kumbhakar SC, Wang H, Horncastle AP (2015). *A practitioner’s guide to stochastic frontier analysis using Stata*. Cambridge University Press.
- Schmidt R, Kneib T (2020). “Analytic expressions for the Cumulative Distribution Function of the Composed Error Term in Stochastic Frontier Analysis with Truncated Normal and Exponential Inefficiencies.” *arXiv preprint arXiv:2006.03459*.

**Examples**

```
### First example with simulated data
#Set seed, sample size and type of function
set.seed(1337)
N=500 #Sample size
s=-1 #Set to production function

#Generate covariates
x1<-runif(N,-1,1); x2<-runif(N,-1,1); x3<-runif(N,-1,1)
x4<-runif(N,-1,1); x5<-runif(N,-1,1)

#Set parameters of the distribution
mu=2+0.75*x1+0.4*x2+0.6*x2^2+6*log(x3+2)^(1/4) #production function parameter
sigma_v=exp(-1.5+0.75*x4) #noise parameter
sigma_u=exp(-1+sin(2*pi*x5)) #inefficiency parameter

#Simulate responses and create dataset
y<-rcomper(n=N, mu=mu, sigma_v=sigma_v, sigma_u=sigma_u, s=s, distr="normhnorm")
dat<-data.frame(y, x1, x2, x3, x4, x5)

#Write formulae for parameters
mu_formula<-y~x1+x2+I(x2^2)+s(x3, bs="ps")
sigma_v_formula<-~1+x4
sigma_u_formula<-~1+s(x5, bs="ps")

#Fit model
model<-dsfa(formula=list(mu_formula, sigma_v_formula, sigma_u_formula),
            data=dat, family=comper(s=s, distr="normhnorm"), optimizer = c("efs"))

#Model summary
summary(model)
```



```

#Smooth effects
#Effect of x3 on the predictor of the production function
plot(model, select=1) #Estimated function
lines(x3[order(x3)], 6*log(x3[order(x3)]+2)^(1/4)-
      mean(6*log(x3[order(x3)]+2)^(1/4)), col=2) #True effect

#Effect of x5 on the predictor of the inefficiency
plot(model, select=2) #Estimated function
lines(x5[order(x5)], -1+sin(2*pi*x5)[order(x5)]-
      mean(-1+sin(2*pi*x5)),col=2) #True effect

### Second example with real data of production function

data("RiceFarms", package = "plm") #load data
RiceFarms[,c("goutput", "size", "seed", "totlabor", "urea")]<-
  log(RiceFarms[,c("goutput", "size", "seed", "totlabor", "urea")]) #log outputs and inputs
RiceFarms$id<-factor(RiceFarms$id) #id as factor

#Set to production function
s=-1

#Write formulae for parameters
mu_formula<-goutput ~ s(size, bs="ps") + s(seed, bs="ps") + #non-linear effects
  s(totlabor, bs="ps") + s(urea, bs="ps") + #non-linear effects
  varieties + #factor
  s(id, bs="re") #random effect
sigma_v_formula<--~1
sigma_u_formula<--~bimas

#Fit model with normhnorm dtribution
model<-dsfa(formula=list(mu_formula, sigma_v_formula, sigma_u_formula),
  data=RiceFarms, family=comper(s=s, distr="normhnorm"), optimizer = "efs")

#Summary of model
summary(model)

#Plot smooths
plot(model)

### Third example with real data of cost function

data("electricity", package = "sfaR") #load data

#Log inputs and outputs as in Greene 1990 eq. 46
electricity$lcof<-log(electricity$cost/electricity$fprice)
electricity$lo<-log(electricity$output)
electricity$llf<-log(electricity$lprice/electricity$fprice)
electricity$lcf<-log(electricity$cprice/electricity$fprice)

#Set to cost function
s=1

#Write formulae for parameters

```

```

mu_formula<-lcof ~ s(lo, bs="ps") + s(llf, bs="ps") + s(lcf, bs="ps") #non-linear effects
sigma_v_formula<-~1
sigma_u_formula<-~s(lo, bs="ps") + s(lshare, bs="ps") + s(cshare, bs="ps")

#Fit model with normhnorm distribution
model<-dsfa(formula=list(mu_formula, sigma_v_formula, sigma_u_formula),
            data=electricity, family=comper(s=s, distr="normhnorm"),
            optimizer = "efs")

#Summary of model
summary(model)

#Plot smooths
plot(model)

```

---

efficiency

*efficiency*


---

## Description

Calculates the expected technical (in)efficiency index.

## Usage

```
efficiency(object, alpha = 0.05, type = "jondrow")
```

## Arguments

object	fitted mgcv object with family <code>comper()</code> or <code>comper_mv()</code> .
alpha	for the $(1 - \alpha) \cdot 100\%$ confidence interval. Must be in $(0,1)$ .
type	default is "jondrow" for $E[U \mathcal{E}]$ , alternatively "battese" for $E[\exp(-U) \mathcal{E}]$ .

## Value

Returns a matrix of the expected (in)efficiency estimates as well the lower and upper bound of the  $(1 - \alpha) \cdot 100\%$  confidence interval.

## References

- Schmidt R, Kneib T (2023). "Multivariate distributional stochastic frontier models." *Computational Statistics & Data Analysis*, 107796.
- Kumbhakar SC, Wang H, Horncastle AP (2015). *A practitioner's guide to stochastic frontier analysis using Stata*. Cambridge University Press.
- Azzalini A (2013). *The skew-normal and related families*, volume 3. Cambridge University Press.

- Jondrow J, Lovell CK, Materov IS, Schmidt P (1982). “On the estimation of technical inefficiency in the stochastic frontier production function model.” *Journal of econometrics*, **19**(2-3), 233–238.
- Battese GE, Coelli TJ (1988). “Prediction of firm-level technical efficiencies with a generalized frontier production function and panel data.” *Journal of econometrics*, **38**(3), 387–399.

## Examples

```
#Set seed, sample size and type of function
set.seed(1337)
N=500 #Sample size
s=-1 #Set to production function

#Generate covariates
x1<-runif(N,-1,1); x2<-runif(N,-1,1); x3<-runif(N,-1,1)
x4<-runif(N,-1,1); x5<-runif(N,-1,1)

#Set parameters of the distribution
mu=2+0.75*x1+0.4*x2+0.6*x2^2+6*log(x3+2)^(1/4) #production function parameter
sigma_v=exp(-1.5+0.75*x4) #noise parameter
sigma_u=exp(-1+sin(2*pi*x5)) #inefficiency parameter

y<-rcomper(n=N, mu=mu, sigma_v=sigma_v, sigma_u=sigma_u, s=s, distr="normhnorm")
dat<-data.frame(y, x1, x2, x3, x4, x5)

#Write formulae for parameters
mu_formula<-y~x1+x2+I(x2^2)+s(x3, bs="ps")
sigma_v_formula<-~1+x4
sigma_u_formula<-~1+s(x5, bs="ps")

#Fit model
model<-dsfa(formula=list(mu_formula, sigma_v_formula, sigma_u_formula),
            data=dat, family=comper(s=s, distr="normhnorm"), optimizer = c("efs"))

#Estimate efficiency
efficiency(model, type="jondrow")
efficiency(model, type="battese")
```

---

elasticity

*elasticity*

---

## Description

Calculates and plots the elasticity of a smooth function.

## Usage

```
elasticity(object, select = NULL, plot = TRUE, se = TRUE)
```

**Arguments**

object	fitted mgcv object with family <code>comper()</code> or <code>comper_mv()</code> .
select	specifying the smooth function for which the elasticity is calculated. If <code>term=NULL</code> the elasticities for all smooths of $\mu$ are returned (excluding random and spatial effects).
plot	logical; if TRUE, plots the elasticities. If FALSE, returns the average elasticity.
se	logical; if TRUE, adds standard errors to the plot of elasticities.

**Details**

Calculates the marginal product for parametric terms. For smooth terms the average of the derivative is calculated.

**Value**

If `plot` is TRUE, plots the elasticities specified in `select` of the provided object. If `plot` is FALSE returns a named vector of the elasticity of the provided inputs.

**References**

- Schmidt R, Kneib T (2023). “Multivariate distributional stochastic frontier models.” *Computational Statistics & Data Analysis*, 107796.
- Kumbhakar SC, Wang H, Horncastle AP (2015). *A practitioner’s guide to stochastic frontier analysis using Stata*. Cambridge University Press.
- Aigner D, Lovell CK, Schmidt P (1977). “Formulation and estimation of stochastic frontier production function models.” *Journal of econometrics*, 6(1), 21–37.
- Meeusen W, van Den Broeck J (1977). “Efficiency estimation from Cobb-Douglas production functions with composed error.” *International economic review*, 435–444.

**Examples**

```
#Set seed, sample size and type of function
set.seed(1337)
N=500 #Sample size
s=-1 #Set to production function

#Generate covariates
x1<-runif(N,-1,1); x2<-runif(N,-1,1); x3<-runif(N,-1,1)
x4<-runif(N,-1,1); x5<-runif(N,-1,1)

#Set parameters of the distribution
mu=2+0.75*x1+0.4*x2+0.6*x2^2+6*log(x3+2)^(1/4) #production function parameter
sigma_v=exp(-1.5+0.75*x4) #noise parameter
sigma_u=exp(-1+sin(2*pi*x5)) #inefficiency parameter

y<-rcomper(n=N, mu=mu, sigma_v=sigma_v, sigma_u=sigma_u, s=s, distr="normhnorm")
dat<-data.frame(y, x1, x2, x3, x4, x5)
```

```

#Write formulae for parameters
mu_formula<-y~x1+x2+I(x2^2)+s(x3, bs="ps")
sigma_v_formula<-~1+x4
sigma_u_formula<-~1+s(x5, bs="ps")

#Fit model
model<-dsfa(formula=list(mu_formula, sigma_v_formula, sigma_u_formula),
            data=dat, family=comper(s=s, distr="normhnorm"), optimizer = c("efs"))

#Get elasticities
elasticity(model, plot=TRUE)

```

ind2joint

*Independent to joint function***Description**

Combines multiple derivs objects into a single derivs object.

**Usage**

```
ind2joint(f_list, tri_f_list, tri_h_list, deriv_order)
```

**Arguments**

f_list	list of derivs objects of length $M$ , e.g. $list(f_1(\cdot), f_2(\cdot), \dots, f_M(\cdot))$
tri_f_list	list of length $K$ trind_generator objects, the $k$ th element corresponds to $k$ th derivs object.
tri_h_list	list of length $K$ trind_generator objects, the $k$ th element corresponds to a derivs object with $k \cdot (k + 1)/2$ parameters.
deriv_order	integer; maximum order of derivative. Available are 0,2 and 4.

**Details**

Let  $f_m$  be a function defined in [trind()], where  $m \in 1, \dots, M$ . Define  $h((x_{n1}, x_{n2}, \dots, x_{nK})) = (f_1(x_{n1}), f_2(x_{n2}), \dots, f_M(x_{nK}))$ . In order to get the derivatives of  $h(\cdot)$  w.r.t all parameters  $x_{nk}$ , the independent functions are combined. For more details see [trind()] and [trind\_generator()].

**Value**

Returns a derivs object.

**See Also**

Other derivs: [chainrule\(\)](#), [derivs\\_transform\(\)](#), [differencerule\(\)](#), [list2derivs\(\)](#), [productrule\(\)](#), [quotientrule\(\)](#), [sumrule\(\)](#), [trind\\_generator\(\)](#), [trind\(\)](#)

**Examples**

```
A<-matrix(c(1:9)/10, ncol=1)
A_derivs<-list2derivs(list(A, A^0, A^2, A^3, A^4), deriv_order=4)
B_derivs<-transform(A, type="exp", par=0, deriv_order=4)
ind2joint (list(A_derivs,B_derivs),
           list(trind_generator(1),trind_generator(1)),
           list(trind_generator(1),trind_generator(1+1)), 4)
```

---

list2derivs

*list2derivs*


---

**Description**

Transforms a list of matrices d0, d1, d2, d3, d4 to a derivs object.

**Usage**

```
list2derivs(f, deriv_order)
```

**Arguments**

f                   list of matrices; d0, d1, d2, d3, d4  
deriv\_order       integer; maximum order of derivative. Available are 0,2 and 4.

**Value**

Mostly internal function. Returns an object of class derivs For more details see [trind()] and [trind\_generator()].

**See Also**

Other derivs: [chainrule\(\)](#), [derivs\\_transform\(\)](#), [differencerule\(\)](#), [ind2joint\(\)](#), [productrule\(\)](#), [quotientrule\(\)](#), [sumrule\(\)](#), [trind\\_generator\(\)](#), [trind\(\)](#)

**Examples**

```
A<-matrix(c(1:9)/10, ncol=3)
list2derivs(list(A, A^0, A^2, A^3, A^4), deriv_order=4)
```

---

manuf

*NBER-CES Manufacturing Dairy Data*

---

### Description

Dataset of the National Bureau of Economic Research (NBER) and U.S. Census Bureau's Center for Economic Studies (CES). It contains information on the annual industry-level from 1958-2016 of the US.

### Usage

manuf

### Format

manuf is a data frame with 6188 rows and 7 columns:

**naics** NAICS 2012 6-digit industry code

**year** Year from 2000 to 2016

**Y** Total value of shipments in millions of 2012 dollars

**K** Real capital stock in millions of 2012 dollars

**L** Production worker hours in millions

**M** Total cost of materials in millions of 2012 dollars

**I** New capital spending in millions of 2012 dollars

### Details

This is a subset of the data which contains data from 2000-2016 on output, employment, materials, investment and capital stocks.

### Source

<<https://www.nber.org/research/data/nber-ces-manufacturing-industry-database>>

### References

Bartlesman E, Gray WB (1996). "The NBER manufacturing productivity database."

mom2par

*Moments to Parameters***Description**

Calculates the parameters of composed-error distribution based on the provided moments.

**Usage**

```
mom2par(mean = 0, sd = 1, skew = 0, s = -1, distr = "normhnorm")
```

**Arguments**

mean	numeric vector of means.
sd	numeric vector of standard deviations. Must be positive.
skew	numeric vector of skewness. $s \cdot \text{skew}$ must be positive.
s	integer; $s = -1$ for production and $s = 1$ for cost function.
distr	string; determines the distribution: 'normhnorm', Normal-halfnormal distribution 'normexp', Normal-exponential distribution

**Details**

See [dcomper](#) for details of the distribution. For the inverse transformation see [par2mom](#).

**Value**

Returns a matrix where the first column corresponds to  $\mu$ , the second to  $\sigma_V$  and the third to  $\sigma_U$ .

**Examples**

```
mom2par(mean=0, sd=1, skew=-0.5, s=-1, distr="normhnorm")
mom2par(mean=0, sd=1, skew=-1, s=-1, distr="normexp")
```



---

par2mom	<i>Parameter to Moments</i>
---------	-----------------------------

---

### Description

Calculates the moments of composed-error distribution based on the provided parameters.

### Usage

```
par2mom(mu = 0, sigma_v = 1, sigma_u = 1, s = -1, distr = "normhnorm")
```

### Arguments

mu	numeric vector of $\mu$ .
sigma_v	numeric vector of $\sigma_V$ . Must be positive.
sigma_u	numeric vector of $\sigma_U$ . Must be positive.
s	integer; $s = -1$ for production and $s = 1$ for cost function.
distr	string; determines the distribution: 'normhnorm', Normal-halfnormal distribution 'normexp', Normal-exponential distribution

### Details

See [dcomper](#) for details of the distribution. For the inverse transformation see [mom2par](#).

### Value

Returns a matrix where the first column corresponds to the mean, the second to the standard deviation and the third to the skewness.

### Examples

```
par2mom(mu=0, sigma_v=1, sigma_u=1, s=-1, distr="normhnorm")
par2mom(mu=0, sigma_v=1, sigma_u=1, s=-1, distr="normexp")
```

---

 productrule

*Productrule*


---

### Description

Productrule for derivs objects.

### Usage

```
productrule(f_list, tri, deriv_order)
```

### Arguments

`f_list` list of derivs objects of length  $M$ , e.g.  $list(f_1(\cdot), f_2(\cdot), \dots, f_M(\cdot))$   
`tri` list; created by the function `[trind_generator()]`.  
`deriv_order` integer; maximum order of derivative. Available are 0,2 and 4.

### Details

Let  $f_m$  be a function defined in `[trind()]`, where  $m \in 1, \dots, M$ . Define  $h((x_{n1}, x_{n2}, \dots, x_{nK})) = f_1(\cdot) \cdot f_2(\cdot) \dots \cdot f_M(x_{n1}, x_{n2}, \dots, x_{nK})$ . In order to get the derivatives of  $h(\cdot)$  w.r.t all parameters  $x_{nk}$ , the productrule is applied. For more details see `[trind()]` and `[trind_generator()]`.

### Value

Returns an object of class `derivs` for the function  $h(\cdot)$ .

### See Also

Other derivs: [chainrule\(\)](#), [derivs\\_transform\(\)](#), [differencerule\(\)](#), [ind2joint\(\)](#), [list2derivs\(\)](#), [quotientrule\(\)](#), [sumrule\(\)](#), [trind\\_generator\(\)](#), [trind\(\)](#)

### Examples

```
A<-matrix(c(1:9)/10, ncol=1)
A_derivs<-list2derivs(list(A, A^0, A^2, A^3, A^4), deriv_order=2)
B_derivs<-derivs_transform(A, type="inv", par=0, trind_generator(1), deriv_order=2)
productrule (list(A_derivs, B_derivs), trind_generator(1), deriv_order=2) #identity
```

---

 quotientrule

*Quotientrule*


---

**Description**

Quotientrule for derivs objects.

**Usage**

```
quotientrule(f_list, tri, deriv_order)
```

**Arguments**

`f_list` list of derivs objects of length  $M$ , e.g. `list(f1(·), f2(·), ..., fM(·))`  
`tri` list; created by the function `[trind_generator()]`.  
`deriv_order` integer; maximum order of derivative. Available are 0,2 and 4.

**Details**

Let  $f_m$  be a function defined in `[trind()]`, where  $m \in 1, \dots, M$ . Define  $h((x_{n1}, x_{n2}, \dots, x_{nK})) = f_1(\cdot)/f_2(\cdot)\dots/f_M(x_{n1}, x_{n2}, \dots, x_{nK})$ . In order to get the derivatives of  $h(\cdot)$  w.r.t all parameters  $x_{nk}$ , the `quotientrule` is applied. For more details see `[trind()]` and `[trind_generator()]`. The values of the derivs objects must be positive. Numerically not precise, but included for reasons of completeness.

**Value**

Returns an object of class `derivs` for the function  $h(\cdot)$ .

**See Also**

Other derivs: [chainrule\(\)](#), [derivs\\_transform\(\)](#), [differencerule\(\)](#), [ind2joint\(\)](#), [list2derivs\(\)](#), [productrule\(\)](#), [sumrule\(\)](#), [trind\\_generator\(\)](#), [trind\(\)](#)

**Examples**

```
A<-matrix(c(1:9)/10, ncol=1)
A_derivs<-list2derivs(list(A, A^0, A^2, A^3, A^4), deriv_order=2)
B_derivs<-derivs_transform(A, type="inv", par=0, trind_generator(1), deriv_order=2)
quotientrule (list(A_derivs, B_derivs), trind_generator(1), deriv_order=2) #A/(1/A)=A^2
```

---

sumrule	<i>Sumrule</i>
---------	----------------

---

### Description

Sumrule for derivs objects.

### Usage

```
sumrule(f_list, tri, deriv_order)
```

### Arguments

f_list	list of derivs objects of length $M$ , e.g. $list(f_1(\cdot), f_2(\cdot), \dots, f_M(\cdot))$
tri	list; created by the function [trind_generator()].
deriv_order	integer; maximum order of derivative. Available are 0,2 and 4.

### Details

Let  $f_m$  be a function defined in [trind()], where  $m \in 1, \dots, M$ . Define  $h((x_{n1}, x_{n2}, \dots, x_{nK})) = f_1(\cdot) + f_2(\cdot) \dots + f_M(x_{n1}, x_{n2}, \dots, x_{nK})$ . In order to get the derivatives of  $h(\cdot)$  w.r.t all parameters  $x_{nk}$ , the sumrule is applied. For more details see [trind()] and [trind\_generator()].

### Value

Returns an object of class derivs for the function  $h(\cdot)$ .

### See Also

Other derivs: [chainrule\(\)](#), [derivs\\_transform\(\)](#), [differencerule\(\)](#), [ind2joint\(\)](#), [list2derivs\(\)](#), [productrule\(\)](#), [quotientrule\(\)](#), [trind\\_generator\(\)](#), [trind\(\)](#)

### Examples

```
A<-matrix(c(1:9)/10, ncol=1)
A_derivs<-list2derivs(list(A, A^0, A^2, A^3, A^4), deriv_order=4)
sumrule(list(A_derivs, A_derivs), trind_generator(1), deriv_order=4) #equal to 2*A_derivs
```

---

transform	<i>transform</i>
-----------	------------------

---

### Description

Transforms a matrix via the specified function.

### Usage

```
transform(x, type, par, deriv_order)
```

### Arguments

x	numeric matrix to be transformed.
type	string, specifies the transformation function. Available are: <ol style="list-style-type: none"> <li>1. 'identity': <math>f(x) = x</math>.</li> <li>2. 'exp': <math>f(x) = \exp\{x\}</math>.</li> <li>3. 'log': <math>f(x) = \log\{x\}</math>.</li> <li>4. 'glogit': <math>f(x) = \log\{(-x + \text{min})/(x - \text{max})\}</math>, where <math>\text{par} = c(\text{min}, \text{max})</math>.</li> <li>5. 'glogitinv': <math>f(x) = \exp\{x\} \cdot (\text{max} + \text{min}) / (1 + \exp\{x\})</math>, where <math>\text{par} = c(\text{min}, \text{max})</math>.</li> <li>6. 'inv': <math>f(x) = \frac{1}{x}</math>.</li> <li>7. 'pnorm': <math>f(x) = \Phi(x)</math>.</li> <li>8. 'qnorm': <math>f(x) = \Phi^{-1}(x)</math>.</li> <li>9. 'mexp': <math>f(x) = -\exp\{x\}</math>.</li> <li>10. 'zeta': <math>f(x) = \log\{2 \cdot \Phi(x)\}</math>.</li> <li>11. 'constant': <math>f(x) = c</math>.</li> <li>12. 'chainrule_utility': <math>f(x) = f'(x) = f''(x) = f'''(x) = f''''(x)</math>.</li> <li>13. onemx: <math>1 - x</math></li> </ol>
par	numeric vector, additional parameters, e.g. min and max for glogit.
deriv_order	integer; maximum order of derivative. Available are 0, 2 and 4.

### Details

Takes the numeric matrix x as an input for the function specified by type and evaluates it together with the derivatives.

### Value

Returns an object of class `derivs`.

### Examples

```
A<-matrix(c(1:9)/10, ncol=3)
A_mat<-list2derivs(list(A, A^0, A^2, A^3, A^4), deriv_order=4)
transform(x=transform(x = A, type="exp", par=0, deriv_order=4), type="log", deriv_order=4, par = 0)
```

---

trind	<i>trind function</i>
-------	-----------------------

---

### Description

Provides the column index of the required derivative for the specified order of a `derivs` object.

### Usage

```
trind(tri, part_deriv_var)
```

### Arguments

`tri` list; created by the function `[trind_generator()]`.

`part_deriv_var` integer vector; specifies  $\frac{\partial^J f(\cdot)}{\partial x_{n i_1} \dots \partial x_{n i_J}}$ . The length of the vector is denoted as  $J$  and determines the order of the partial derivatives with maximum four. The element  $i_j \in \{0, \dots, K - 1\}$  specifies the variable with respect to which the derivative is taken, where  $j \in \{1, \dots, J\}$ , The order corresponds to the order of derivatives. For example `c(0, 0, 1, 2)` is equal to  $\frac{\partial^4 f(\cdot)}{\partial x_{n1} \partial x_{n1} \partial x_{n2} \partial x_{n3}}$ . See details for more information.

### Details

Let  $f : \mathbb{R}^K \rightarrow \mathbb{R}^L, (x_{n1}, x_{n2}, \dots, x_{nK}) \rightarrow f(x_{n1}, x_{n2}, \dots, x_{nK})$  be differentiable up to order four w.r.t all parameters  $x_{nk}$ , where  $k \in \{1, \dots, K\}$  and  $n \in \{1, \dots, N\}$ . Then a `derivs` class object is a numeric matrix with  $N$  rows and  $L$  columns.  $N$  is the length of the input vectors. Further, it has the following attributes:

1. ‘d1’: a numeric matrix of the first derivatives w.r.t all parameters, where the  $n$ th row corresponds to:  $(\frac{\partial f(\cdot)}{\partial x_{n1}}, \frac{\partial f(\cdot)}{\partial x_{n1}}, \dots, \frac{\partial f(\cdot)}{\partial x_{nK}})$
2. ‘d2’: a numeric matrix of the second derivatives w.r.t all parameters, where the  $n$ th row corresponds to:  $(\frac{\partial^2 f(\cdot)}{\partial x_{n1} \partial x_{n1}}, \frac{\partial^2 f(\cdot)}{\partial x_{n1} \partial x_{n2}}, \dots, \frac{\partial^2 f(\cdot)}{\partial x_{nK} \partial x_{nK}})$
3. ‘d3’: a numeric matrix of the third derivatives w.r.t all parameters, where the  $n$ th row corresponds to:  $(\frac{\partial^3 f(\cdot)}{\partial x_{n1} \partial x_{n1} \partial x_{n1}}, \frac{\partial^3 f(\cdot)}{\partial x_{n1} \partial x_{n1} \partial x_{n2}}, \dots, \frac{\partial^3 f(\cdot)}{\partial x_{nK} \partial x_{nK} \partial x_{nK}})$
4. ‘d4’: a numeric matrix of the fourth derivatives w.r.t all parameters, where the  $n$ th row corresponds to:  $(\frac{\partial^4 f(\cdot)}{\partial x_{n1} \partial x_{n1} \partial x_{n1} \partial x_{n1}}, \frac{\partial^4 f(\cdot)}{\partial x_{n1} \partial x_{n1} \partial x_{n1} \partial x_{n2}}, \dots, \frac{\partial^4 f(\cdot)}{\partial x_{nK} \partial x_{nK} \partial x_{nK} \partial x_{nK}})$

The function `trind()` provides the index for the corresponding derivatives. The `derivs` class object allows for a modular system which can be easily extended and is faster than numerical derivatives. The advantage compared to analytical derivatives provided by ‘`mathematica`’ or `deriv()` is that asymptotics and approximations can be used for individual parts. Handwritten derivatives can be tedious at times and may be prone to errors. Thus, the `derivs` class object can be used by lazy users. Mainly intended for internal use.

**Value**

Integer, the index for a derivs object.

**See Also**

Other derivs: [chainrule\(\)](#), [derivs\\_transform\(\)](#), [differencerule\(\)](#), [ind2joint\(\)](#), [list2derivs\(\)](#), [productrule\(\)](#), [quotientrule\(\)](#), [sumrule\(\)](#), [trind\\_generator\(\)](#)

**Examples**

```
tri=trind_generator(3)
trind(tri, c(2,1))
```

---

trind_generator	<i>Trind_generator function</i>
-----------------	---------------------------------

---

**Description**

Generates index matrices for upper triangular storage up to order four.

**Usage**

```
trind_generator(K)
```

**Arguments**

K                    integer; determines the number of parameters.

**Details**

Useful when working with higher order derivatives, which generate symmetric arrays. Mainly intended for internal use. Similar to 'mgcv::trind.generator'. Mostly internal function.

**Value**

Returns a list with index matrices for the first to fourth derivative, which can be accessed via the function [trind()]. The numerical vectors `i_start` and `i_end` hold the starting and ending indexes, which are required by [trind()] for derivatives greater than two.

**See Also**

Other derivs: [chainrule\(\)](#), [derivs\\_transform\(\)](#), [differencerule\(\)](#), [ind2joint\(\)](#), [list2derivs\(\)](#), [productrule\(\)](#), [quotientrule\(\)](#), [sumrule\(\)](#), [trind\(\)](#)

**Examples**

```
tri<-trind_generator(3)
tri_mgcv<-mgcv::trind.generator(3)

for(i in 1:3){
  print(i==trind(tri, part_deriv_var=c(i)-1)+1)
  for(j in i:3){
    print(tri_mgcv$i2[i,j]==trind(tri, part_deriv_var=c(i,j)-1)+1)
    for(k in j:3){
      print(tri_mgcv$i3[i,j,k]==trind(tri, part_deriv_var=c(i,j,k)-1)+1)
      for(l in k:3){
        print(tri_mgcv$i4[i,j,k,l]==trind(tri, part_deriv_var=c(i,j,k,l)-1)+1)
      }
    }
  }
}
```



# Index

- \* **copula**
  - cop, 13
  - dcop, 20
  - delta\_bounds, 22
- \* **datasets**
  - BurkinaFarms, 2
  - BurkinaFarms\_polys, 4
  - manuf, 39
- \* **derivs**
  - chainrule, 5
  - derivs\_transform, 23
  - differencerule, 24
  - ind2joint, 37
  - list2derivs, 38
  - productrule, 42
  - quotientrule, 43
  - sumrule, 44
  - trind, 46
  - trind\_generator, 47
- \* **distribution**
  - dcomper, 15
  - dcomper\_mv, 17
  - dnormexp, 25
  - dnormhnorm, 27
- anova.gam, 30
- BurkinaFarms, 2
- BurkinaFarms\_polys, 4
- cdf2quantile, 4
- chainrule, 5, 24, 25, 37, 38, 42–44, 47
- comper, 6, 30, 31
- comper\_mv, 9, 30, 31
- cop, 13, 22, 23, 31
- dcomper, 15, 20, 27, 29–31, 40, 41
- dcomper\_mv, 17, 17, 27, 29, 31
- dcop, 14, 20, 23, 31
- delta\_bounds, 14, 22, 22
- deriv(), 46
- derivs\_transform, 5, 23, 25, 37, 38, 42–44, 47
- differencerule, 5, 24, 24, 37, 38, 42–44, 47
- dnormexp, 16, 17, 20, 25, 29, 30
- dnormhnorm, 16, 17, 20, 27, 27, 30
- dsfa, 29
- efficiency, 30, 34
- elasticity, 30, 35
- formula.gam, 31
- gam, 30, 31
- gam(), 6, 10, 14
- gam.models, 31
- gamObject, 30
- ind2joint, 5, 24, 25, 37, 38, 42–44, 47
- list2derivs, 5, 24, 25, 37, 38, 42–44, 47
- manuf, 39
- mom2par, 40, 41
- mrf, 30
- normexp (dnormexp), 25
- normhnorm (dnormhnorm), 27
- p.spline, 30
- par2mom, 40, 41
- pcomper (dcomper), 15
- pcomper\_mv (dcomper\_mv), 17
- pcop (dcop), 20
- pCopula, 21
- plot.gam, 30
- pnormexp (dnormexp), 25
- pnormhnorm (dnormhnorm), 27
- predict.gam, 30
- productrule, 5, 24, 25, 37, 38, 42, 43, 44, 47
- qcomper (dcomper), 15

qnormexp (dnormexp), 25  
qnormhnorm (dnormhnorm), 27  
quotientrule, 5, 24, 25, 37, 38, 42, 43, 44, 47

random.effects, 30  
rcomper (dcomper), 15  
rcomper\_mv (dcomper\_mv), 17  
rcop (dcop), 20  
rCopula, 21  
residuals.gam, 30  
rnormexp (dnormexp), 25  
rnormhnorm (dnormhnorm), 27

s, 30  
smooth.terms, 30  
summary.gam, 30  
sumrule, 5, 24, 25, 37, 38, 42, 43, 44, 47

tprs, 30  
transform, 45  
trind, 5, 21, 24–26, 28, 37, 38, 42–44, 46, 47  
trind\_generator, 5, 16, 21, 24–26, 28, 37,  
38, 42–44, 47, 47