

# Package: GCPM (via r-universe)

September 12, 2024

**Type** Package

**Title** Generalized Credit Portfolio Model

**Version** 1.2.2

**Date** 2016-12-29

**Author** Kevin Jakob

**Maintainer** Kevin Jakob <Kevin.Jakob.Research@gmail.com>

**Description** Analyze the default risk of credit portfolios. Commonly known models, like CreditRisk+ or the CreditMetrics model are implemented in their very basic settings. The portfolio loss distribution can be achieved either by simulation or analytically in case of the classic CreditRisk+ model. Models are only implemented to respect losses caused by defaults, i.e. migration risk is not included. The package structure is kept flexible especially with respect to distributional assumptions in order to quantify the sensitivity of risk figures with respect to several assumptions. Therefore the package can be used to determine the credit risk of a given portfolio as well as to quantify model sensitivities.

**License** GPL-2

**Imports** Rcpp (>= 0.11.2), methods, RcppProgress(>= 0.1), parallel

**LinkingTo** Rcpp, RcppProgress

**SystemRequirements** Windows, Linux, OS X

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2016-12-30 00:34:04

## Contents

GCPM-package . . . . .	3
alpha.max-methods . . . . .	4
analyze-methods . . . . .	5

business-methods . . . . .	7
CDF-methods . . . . .	8
country-methods . . . . .	9
default-methods . . . . .	9
EAD-methods . . . . .	10
EC-methods . . . . .	10
EC.cont-methods . . . . .	11
EL-methods . . . . .	11
EL.analyt-methods . . . . .	12
ES-methods . . . . .	13
ES.cont-methods . . . . .	13
export-methods . . . . .	14
GCPM-class . . . . .	14
idiosyncr-methods . . . . .	17
init . . . . .	17
LGD-methods . . . . .	19
LHR-methods . . . . .	20
link.function-methods . . . . .	21
loss-methods . . . . .	21
loss.thr-methods . . . . .	22
loss.unit-methods . . . . .	22
model.type-methods . . . . .	23
N-methods . . . . .	23
name-methods . . . . .	24
NC-methods . . . . .	24
NR-methods . . . . .	25
NS-methods . . . . .	25
PD-methods . . . . .	26
PDF-methods . . . . .	26
PL-methods . . . . .	27
plot-methods . . . . .	27
portfolio.pois . . . . .	28
portfolio.pool . . . . .	28
portfolios . . . . .	29
random.numbers-methods . . . . .	30
SD-methods . . . . .	30
SD.analyt-methods . . . . .	31
SD.cont-methods . . . . .	31
SD.div-methods . . . . .	32
SD.syst-methods . . . . .	32
sec.var-methods . . . . .	33
sector.names-methods . . . . .	33
seed-methods . . . . .	34
show-methods . . . . .	34
summary-methods . . . . .	34
VaR-methods . . . . .	35
VaR.cont-methods . . . . .	35
W-methods . . . . .	36

---

GCPM-package

*Generalized Credit Portfolio Model*

---

## Description

The package helps to analyze the default risk of credit portfolios. Commonly known models, like CreditRisk+ or the CreditMetrics model are implemented in their very basic settings. The portfolio loss distribution can be achieved either by simulation or analytically in case of the classic CreditRisk+ model. Models are only implemented to respect losses caused by defaults, i.e. migration risk is not included. The package structure is kept flexible especially with respect to distributional assumptions in order to quantify the sensitivity of risk figures with respect to several assumptions. Therefore the package can be used to determine the credit risk of a given portfolio as well as to quantify model sensitivities.

## Details

Package: GCPM  
Type: Package  
Version: 1.2.2  
Date: 2016-12-29  
License: GPL-2

## Author(s)

Kevin Jakob

Maintainer: Kevin Jakob <Kevin.Jakob.Research@gmail.com>

## References

Jakob, K. & Fischer, M. "GCPM: A flexible package to explore credit portfolio risk" *Austrian Journal of Statistics* 45.1 (2016): 25:44  
Morgan, J. P. "CreditMetrics-technical document." JP Morgan, New York, 1997  
First Boston Financial Products, "CreditRisk+", 1997  
Gundlach & Lehrbass, "CreditRisk+ in the Banking Industry", Springer, 2003

## See Also

[GCPM-class](#), [init](#), [analyze](#)

**Examples**

```

#create a random portfolio with NC counterparties
NC=100
#assign business lines and countries randomly
business.lines=c("A","B","C")
CP.business=business.lines[ceiling(runif(NC,0,length(business.lines)))]
countries=c("A","B","C","D","E")
CP.country=countries[ceiling(runif(NC,0,length(countries)))]

#create matrix with sector weights (CreditRisk+ setting)
#according to business lines
NS=length(business.lines)
W=matrix(0,nrow = NC,ncol = length(business.lines),
dimnames = list(1:NC,business.lines))
for(i in 1:NC){W[i,CP.business[i]]=1}

#create portfolio data frame
portfolio=data.frame(Number=1:NC,Name=paste("Name ",1:NC),Business=CP.business,
Country=CP.country,EAD=runif(NC,1e3,1e6),LGD=runif(NC),
PD=runif(NC,0,0.3),Default=rep("Bernoulli",NC),W)

#draw sector variances randomly
sec.var=runif(NS,0.5,1.5)
names(sec.var)=business.lines

#draw N sector realizations (independent gamma distributed sectors)
N=5e4
random.numbers=matrix(NA,ncol=NS,nrow=N,dimnames=list(1:N,business.lines))
for(i in 1:NS){
random.numbers[,i]=rgamma(N,shape = 1/sec.var[i],scale=sec.var[i])}

#create a portfolio model and analyze the portfolio
TestModel=init(model.type = "simulative",link.function = "CRP",N = N,
loss.unit = 1e3, random.numbers = random.numbers,LHR=rep(1,N),loss.thr=5e6,
max.entries=2e4)
TestModel=analyze(TestModel,portfolio)

#plot of pdf of portfolio loss (in million) with indicators for EL, VaR and ES
alpha=c(0.995,0.999)
plot(TestModel,1e6,alpha=alpha)

#calculate portfolio VaR and ES
VaR=VaR(TestModel,alpha)
ES=ES(TestModel,alpha)

#Calculate risk contributions to VaR and ES
risk.cont=cbind(VaR.cont(TestModel,alpha = alpha),
ES.cont(TestModel,alpha = alpha))

```

**Description**

Get the maximum value of the model's CDF. For simulative models, the value should be equal to 1. For an analytical model, the value depends on the value specified during initiation of the model (see [init](#)).

**Usage**

```
alpha.max(this)
```

**Arguments**

`this`                    Object of class GCPM

**Value**

numeric of length 1

**See Also**

[init](#)

---

analyze-methods

*Analyze a Credit Portfolio*

---

**Description**

The method analyzes a given portfolio with a predefined portfolio model (i.e. a GCPM object). Portfolio key numbers such as the number of portfolio positions, sum of EAD and PL or the expected loss are calculated. Afterwards the loss distribution is estimated according to `model.type`.

**Usage**

```
analyze(this,portfolio,alpha,Ncores)
```

**Arguments**

`this`                    object of class GCPM

`portfolio`                data frame containing portfolio data. The following columns have to be defined (please be aware of the correct spelling of the column names):

- Number: identification number for each portfolio position (numeric)
- Name: counterparty name (character)
- Business: business information (character/factor)
- Country: country information (character/factor)
- EAD: exposure at default (numeric)
- LGD: loss given default (numeric in [0,1])
- PD: probability of default (numeric in [0,1])
- Default: default distribution either "Bernoulli" or "Poisson" (employable for pools)
- sectors: starting with the 9th column, the sector weights have to be defined..

alpha	loss levels for risk measures economic capital, value at risk and expected short-fall (optional)
Ncores	number of (virtual) cores used to perform Monte Carlo simulation (requires package <b>parallel</b> , default=1)

### Details

In case of an analytical CreditRisk+ model, a modified version of the algorithm described in Gundlach & Lehrbass (2003) is used. For a simulative model, the loss distribution is estimated based on  $N$  simulations with sector drawings specified by `random.numbers` (see `init`). The sector names (column names) should not include any white spaces. In case of a CreditMetrics type model, the values of  $R$  (not  $R^2$ ) have to be provided as sector weights. In the standard CreditMetrics or CreditRisk+ framework a counterparty can be assigned to more than one sector. Within a analytical CreditRisk+ model, the sector names have to match the names of `sec.var` or in a simulative model the column names of `random.numbers` (see `init`)

### Value

object of class GCPM.

### Methods

`signature(this = "GCPM", portfolio = "data.frame", alpha = "missing")` If loss levels alpha are not provided, risk measures such as economic capital, value at risk and expected shortfall are not calculated by default. However, they can be calculated afterwards by calling the corresponding methods (see `VaR`, `ES`, `EC`)

`signature(this = "GCPM", portfolio = "data.frame", alpha = "numeric")` If loss levels alpha are provided, risk measures such as economic capital, value at risk and expected shortfall are calculated and printed. To extract these risk measures into a separate variable you can use the corresponding methods.

### References

- Jakob, K. & Fischer, M. "GCPM: A flexible package to explore credit portfolio risk" *Austrian Journal of Statistics* 45.1 (2016): 25:44  
 Morgan, J. P. "CreditMetrics-technical document." JP Morgan, New York, 1997  
 First Boston Financial Products, "CreditRisk+", 1997  
 Gundlach & Lehrbass, "CreditRisk+ in the Banking Industry", Springer, 2003

### See Also

`init`

### Examples

```
#create a random portfolio with NC counterparties
NC=100
#assign business lines and countries randomly
business.lines=c("A","B","C")
```

```

CP.business=business.lines[ceiling(runif(NC,0,length(business.lines)))]
countries=c("A","B","C","D","E")
CP.country=countries[ceiling(runif(NC,0,length(countries)))]

#create matrix with sector weights (CreditRisk+ setting)
#according to business lines
NS=length(business.lines)
W=matrix(0,nrow = NC,ncol = length(business.lines),
dimnames = list(1:NC,business.lines))
for(i in 1:NC){W[i,CP.business[i]]=1}

#create portfolio data frame
portfolio=data.frame(Number=1:NC,Name=paste("Name ",1:NC),Business=CP.business,
Country=CP.country,EAD=runif(NC,1e3,1e6),LGD=runif(NC),
PD=runif(NC,0,0.3),Default=rep("Bernoulli",NC),W)

#draw sector variances randomly
sec.var=runif(NS,0.5,1.5)
names(sec.var)=business.lines

#draw N sector realizations (independent gamma distributed sectors)
N=5e4
random.numbers=matrix(NA,ncol=NS,nrow=N,dimnames=list(1:N,business.lines))
for(i in 1:NS){
random.numbers[,i]=rgamma(N,shape = 1/sec.var[i],scale=sec.var[i])}

#create a portfolio model and analyze the portfolio
TestModel=init(model.type = "simulative",link.function = "CRP",N = N,
loss.unit = 1e3, random.numbers = random.numbers,LHR=rep(1,N),loss.thr=5e6,
max.entries=2e4)
TestModel=analyze(TestModel,portfolio)

#plot of pdf of portfolio loss (in million) with indicators for EL, VaR and ES
alpha=c(0.995,0.999)
plot(TestModel,1e6,alpha=alpha)

#calculate portfolio VaR and ES
VaR=VaR(TestModel,alpha)
ES=ES(TestModel,alpha)

#Calculate risk contributions to VaR and ES
risk.cont=cbind(VaR.cont(TestModel,alpha = alpha),
ES.cont(TestModel,alpha = alpha))

#Use parallel computing for Monte Carlo simulation
TestModel=analyze(TestModel,portfolio,Ncores=2)

```

**Description**

Get the business information for each counterparty defined in the portfolio.

**Usage**

```
business(this)
```

**Arguments**

this            Object of class GCPM

**Value**

factor of length equal to number of portfolio positions

**See Also**

[portfolio.pois](#)

---

CDF-methods

*Cumulative Distribution Function of Portfolio Loss*

---

**Description**

Get the CDF of the portfolio loss, available after execution of `analyze`.

**Usage**

```
CDF(this)
```

**Arguments**

this            Object of class GCPM

**Value**

numeric vector

**See Also**

[analyze](#)



---

country-methods	<i>Country Information</i>
-----------------	----------------------------

---

**Description**

Get the country information of each counterparty defined in the portfolio.

**Usage**

```
country(this)
```

**Arguments**

this	Object of class GCPM
------	----------------------

**Value**

factor of length equal to number of portfolio positions

**See Also**

[portfolio.pois](#)

---

default-methods	<i>Default Distribution</i>
-----------------	-----------------------------

---

**Description**

Get the default distribution of each portfolio position. Using “Poisson” as default distribution one can simulate the standard CR+ model or group smaller counterparties into a pool and simulate their defaults.

**Usage**

```
default(this)
```

**Arguments**

this	Object of class GCPM
------	----------------------

**Value**

character of length equal to number of portfolio positions

**See Also**

[portfolio.pois](#)

---

EAD-methods

*Exposure at Default*

---

**Description**

Get the counterparties' exposure at default defined in the portfolio data.

**Usage**

EAD(this)

**Arguments**

this            Object of class GCPM

**Value**

numeric value of length equal to the number of counterparties

**See Also**

[portfolio.pois](#)

---

EC-methods

*Economic Capital*

---

**Description**

Get the value of economic capital for the portfolio on level(s) alpha

**Usage**

EC(this, alpha)

**Arguments**

this            Object of class GCPM  
alpha            numeric vector of loss levels between 0 and 1

**Value**

numeric vector of length equal to length(alpha).

**Description**

Calculate contributions to the economic capital on portfolio level for each portfolio position. In case of a simulative model, the risk contributions are calculated as contributions to expected shortfall on a lower loss level  $\tau$ , such that  $ES(\tau)$  is as close as possible to  $EC(\alpha)$ . Furthermore, in case of a simulative model, loss scenarios above a predefined threshold (`loss.thr`) are analyzed in order to calculate the risk contributions. If `loss.thr` is too high (depending on value of alpha) the calculation will be not possible.

**Usage**

```
EC.cont(this,alpha)
```

**Arguments**

<code>this</code>	Object of class GCPM
<code>alpha</code>	numeric vector of loss levels between 0 and 1

**Value**

numeric matrix with number of rows equal to number of counterparties within the portfolio and number of columns equal to `length(alpha)`

**See Also**

[loss.thr](#)

**Description**

Get the expected loss (EL) calculated from the portfolio loss distribution. Because of the discretization and/or simulation errors, this is not equal to the analytical EL (see [EL.analyt](#)). Please also note, that in case of a simulative model (with Bernoulli default distribution) of the CreditRisk+ type the simulated EL tends to be smaller than the analytical one because the conditional  $\overline{PD} = PD \cdot (w^T x)$  has to be truncated (if  $\overline{PD} > 1$ ).

**Usage**

```
EL(this)
```

**Arguments**

`this`            Object of class GCPM

**Value**

numeric value of length 1

**See Also**

[EL.analyt](#)

---

`EL.analyt-methods`        *Expected Loss (analytical)*

---

**Description**

Get the expected loss (EL) calculated from the portfolio data. Because of the discretization and/or simulation errors, this is not equal to the EL calculated from the portfolio loss distribution (see [EL](#)).

**Usage**

```
EL.analyt(this)
```

**Arguments**

`this`            Object of class GCPM

**Value**

numeric value of length 1

**See Also**

[EL](#)

---

ES-methods                      *Expected Shortfall*

---

**Description**

Get the value of the expected shortfall for the portfolio on level(s) alpha

**Usage**

```
ES(this,alpha)
```

**Arguments**

this	Object of class GCPM
alpha	numeric vector of loss levels between 0 and 1

**Value**

numeric vector of length equal to length(alpha).

---

ES.cont-methods                      *Risk Contributions to Expected Shortfall*

---

**Description**

Calculate contributions to the expected shortfall on portfolio level for each portfolio position. In case of a simulative model, loss scenarios above a predefined threshold (`loss.thr`) are analyzed in order to calculate the risk contributions. If `loss.thr` is too high, calculation may be not possible (depending on value of alpha).

**Usage**

```
ES.cont(this,alpha)
```

**Arguments**

this	Object of class GCPM
alpha	numeric vector of loss levels between 0 and 1

**Value**

numeric matrix with number of rows equal to number of counterparties within the portfolio and number of columns equal to length(alpha)

**See Also**

[loss.thr](#)

---

export-methods	<i>Export Main Results</i>
----------------	----------------------------

---

### Description

This method provides an easy way to export the main results of the portfolio (i.e. after running analyze). A summary file and the portfolio loss distribution (PDF and CDF) are exported to path.out. With the help of file.format one can specify the csv format (“csv1” or “csv2”). If a vector alpha of loss levels is specified, risk contributions to EC, VaR and ES are also exported according to level(s) alpha.

### Usage

```
export(this,path.out,file.format,alpha)
```

### Arguments

this	Object of class GCPM
path.out	string specifying the output path
file.format	string specifying the file format (i.e “csv1” or “csv2”)
alpha	numeric vector with loss levels between 0 and 1

---

GCPM-class	<i>Class "GCPM"</i>
------------	---------------------

---

### Description

The class represents a generalized credit portfolio framework. Users which are not familiar with credit portfolio models in general and the CreditRisk+ model as well as the CreditMetrics model in particular should refer to the references given below. Models can be simulative or analytical (in case of a CreditRisk+ type model). The link function can be chosen to be either of the CreditRisk+ or the CreditMetrics type. Counterparties' default distribution can be specified to be either Bernoulli or Poisson, which is the default distribution in the basic CreditRisk+ framework.

### Objects from the Class

Objects can be created via the `init` function (see [init](#))

**Slots**

- model.type:** Character value, specifying the model type. One can choose between “simulative” and “CRP” which corresponds to the analytical version of the CreditRisk+ model (see First Boston Financial Products, 1997)
- default:** Character vector specifying the counterparties’ default distribution (either “Bernoulli” or “Poisson”)
- link.function:** character value, specifying the type of the link function. One can choose between “CRP”, which corresponds to  $\overline{PD} = PD \cdot (w^T x)$  and “CM” which corresponds to  $\overline{PD} = \Phi \left( \frac{\Phi^{-1} PD - w^T x}{\sqrt{1 - w^T \Sigma w}} \right)$ , where PD is the original PD from portfolio data, x is the vector of sector drawings,  $\Phi$  is the CDF of the standard normal distribution, w is the vector of sector weights given in the portfolio data and  $\Sigma$  is the correlation matrix of the sector variables estimated from random numbers. “CRP” will be used automatically if `model.type == "CRP"`.
- loss.unit:** numeric value used to discretize potential losses.
- NS:** number of sectors
- NC:** number of counterparties
- name:** counterparties’ names defined in the portfolio
- NR:** counterparties’ identification numbers defined in the portfolio
- EAD:** counterparties’ exposure at default defined in the portfolio
- LGD:** counterparties’ loss given default defined in the portfolio
- PL:** counterparties’ potential loss ( $EAD * LGD$ )
- PD:** counterparties’ probability of default defined in the portfolio
- business:** counterparties’ business line defined in the portfolio
- country:** counterparties’ country defined in the portfolio
- EL.analyt:** Expected loss calculated from portfolio data (without discretization)
- EL:** Expected loss derived from loss distribution
- nu:** multiples of loss unit representing discretized potential losses within an analytical CreditRisk+ type model
- PL.disc:** counterparties’ potential loss ( $EAD * LGD$ ) after discretization
- PD.disc:** counterparties’ probability of default defined in the portfolio after discretization
- sec.var:** sector variances within an analytical CreditRisk+ type model
- sector.names:** sector names
- SD.div:** diversifiable part of portfolio risk (measured by standard deviation) in case of a CreditRisk+ type model
- SD.syst:** Non-diversifiable part of portfolio risk (measured by standard deviation) in case of a CreditRisk+ type model
- SD.analyt:** portfolio standard deviation derived from portfolio data in case of a CreditRisk+ type model
- SD:** portfolio standard deviation derived from loss distribution
- W:** counterparties’ sector weights

`idiosyncr`: counterparties idiosyncratic weight in case of a CreditRisk+ type model  
`alpha.max`: maximum level of CDF of the loss distribution within an analytical CreditRisk+ type model  
`a`: internal parameter used to calculate risk contributions in case of an analytical CreditRisk+ type model  
`PDF`: probability density function of portfolio losses  
`CDF`: cumulative distribution function of portfolio losses  
`B`: internal parameter used to calculate risk contributions in case of an analytical CreditRisk+ type model  
`loss`: portfolio losses corresponding to PDF and CDF  
`random.numbers`: sector drawing in case of a simulative model  
`LHR`: likelihood ration of sector drawing in case of a simulative model  
**max.entries** numeric value defining the maximum number of loss scenarios stored to calculate risk contributions.  
`N`: number of simulations in case of a simulative model  
`scenarios`: scenarios (rows) of `random.numbers` used within the simulation of portfolio losses  
`seed`: parameter used to initialize the random number generator. If `seed` is not provided a value based on current system time will be used.  
`loss.thr`: specifies a lower bound for portfolio losses to be stored in order to derive risk contributions on counterparty level. Using a lower value needs a lot of memory but will be necessary in order to calculate risk contributions on lower CDF levels. This parameter is used only if `model.type == "simulative"`.  
`sim.losses`: simulated portfolio losses in case of a simulative model  
`CP.sim.losses`: simulated losses on counterparty level when the overall portfolio loss is greater or equal to `loss.thr`

### Author(s)

Kevin Jakob

### References

Jakob, K. & Fischer, M. "GCPM: A flexible package to explore credit portfolio risk" *Austrian Journal of Statistics* 45.1 (2016): 25:44  
 Morgan, J. P. "CreditMetrics-technical document." JP Morgan, New York, 1997  
 First Boston Financial Products, "CreditRisk+", 1997  
 Gundlach & Lehrbass, "CreditRisk+ in the Banking Industry", Springer, 2003

### See Also

[GCPM-package](#), [init](#), [analyze](#)



---

idiosyncr-methods      *Idiosyncratic Risk Weights*

---

**Description**

Get the idiosyncratic risk weights (i.e. risk weights which are not assigned to any sector). Currently only available if `model.type == "CRP"`.

**Usage**

```
idiosyncr(this)
```

**Arguments**

`this`                  Object of class GCPM

**Value**

numeric vector of length equal to number of counterparties

---

`init`                      *Initialize an Object of Class GCPM*

---

**Description**

The function helps to create a new object of class GCPM. The arguments of the function are passed to the object after performing some plausibility checks.

**Usage**

```
init(model.type = "CRP", link.function = "CRP", N, seed,
      loss.unit, alpha.max = 0.9999, loss.thr = Inf, sec.var,
      random.numbers = matrix(), LHR, max.entries=1e3)
```

**Arguments**

`model.type`              Character value, specifying the model type. One can choose between “simulative” and “CRP” which corresponds to the analytical version of the CreditRisk+ model (see First Boston Financial Products, 1997)

`link.function`          character value, specifying the type of the link function. One can choose between “CRP”, which corresponds to  $\overline{PD} = PD \cdot (w^T x)$  and “CM” which corresponds to  $\overline{PD} = \Phi\left(\frac{\Phi^{-1}PD - w^T x}{\sqrt{1 - w^T \Sigma w}}\right)$ , where PD is the original PD from portfolio data,  $x$  is the vector of sector drawings,  $\Phi$  is the CDF of the standard normal distribution,  $w$  is the vector of sector weights given in the portfolio data and  $\Sigma$  is the correlation matrix of the sector variables estimated from `random.numbers`. “CRP” will be used automatically if `model.type == "CRP"`.

N	numeric value, defining the number of simulations if <code>model.type == "simulative"</code> . If N is greater than the number of scenarios provided via <code>random.numbers</code> , scenarios are reused. This parameter is used only if <code>model.type == "simulative"</code> .
seed	numeric value used to initialize the random number generator. If seed is not provided a value based on current system time will be used. This parameter is used only if <code>model.type == "simulative"</code> .
loss.unit	numeric positive value used to discretize potential losses.
alpha.max	numeric value between 0 and 1 defining the maximum CDF-level which will be computed in case of an analytical CreditRisk+ type model.
loss.thr	numeric value specifying a lower bound for portfolio losses to be stored in order to derive risk contributions on counterparty level. Using a lower value needs a lot of memory but will be necessary in order to calculate risk contributions on lower CDF levels. This parameter is used only if <code>model.type == "simulative"</code> .
sec.var	named numeric vector defining the sector variances in case of a CreditRisk+ type model. The names have to correspond to the sector names given in the portfolio. This parameter is used only if <code>model.type == "CRP"</code> .
random.numbers	matrix with sector drawings. The columns represent the sectors, whereas the rows represent the scenarios (number of different simulations). The column names must correspond to the names used in the portfolio data (see <a href="#">analyze</a> ) and to the names of <code>sec.var</code> if <code>model.type == "CRP"</code> . This parameter is used only if <code>model.type == "simulative"</code> .
LHR	numeric vector of length equal to <code>nrow(random.numbers)</code> defining the likelihood ratio of each scenario. If not provided, all scenarios are assumed to be equally likely. This parameter is used only if <code>model.type == "simulative"</code> .
max.entries	numeric value defining the maximum number of loss scenarios stored to calculate risk contributions.

**Value**

object of class GCPM

**Author(s)**

Kevin Jakob

**References**

Jakob, K. & Fischer, M. "GCPM: A flexible package to explore credit portfolio risk" *Austrian Journal of Statistics* 45.1 (2016): 25:44  
 Morgan, J. P. "CreditMetrics-technical document." JP Morgan, New York, 1997  
 First Boston Financial Products, "CreditRisk+", 1997  
 Gundlach & Lehrbass, "CreditRisk+ in the Banking Industry", Springer, 2003

**See Also**

[GCPM](#), [GCPM-class](#), [analyze](#)

**Examples**

```

#create a random portfolio with NC counterparties
NC=100
#assign business lines and countries randomly
business.lines=c("A","B","C")
CP.business=business.lines[ceiling(runif(NC,0,length(business.lines)))]
countries=c("A","B","C","D","E")
CP.country=countries[ceiling(runif(NC,0,length(countries)))]

#create matrix with sector weights (CreditRisk+ setting)
#according to business lines
NS=length(business.lines)
W=matrix(0,nrow = NC,ncol = length(business.lines),
dimnames = list(1:NC,business.lines))
for(i in 1:NC){W[i,CP.business[i]]=1}

#create portfolio data frame
portfolio=data.frame(Number=1:NC,Name=paste("Name ",1:NC),Business=CP.business,
Country=CP.country,EAD=runif(NC,1e3,1e6),LGD=runif(NC),
PD=runif(NC,0,0.3),Default=rep("Bernoulli",NC),W)

#draw sector variances randomly
sec.var=runif(NS,0.5,1.5)
names(sec.var)=business.lines

#draw N sector realizations (independent gamma distributed sectors)
N=5e4
random.numbers=matrix(NA,ncol=NS,nrow=N,dimnames=list(1:N,business.lines))
for(i in 1:NS){
random.numbers[,i]=rgamma(N,shape = 1/sec.var[i],scale=sec.var[i])}

#create a portfolio model and analyze the portfolio
TestModel=init(model.type = "simulative",link.function = "CRP",N = N,
loss.unit = 1e3, random.numbers = random.numbers,LHR=rep(1,N),loss.thr=5e6,
max.entries=2e4)
TestModel=analyze(TestModel,portfolio)

#plot of pdf of portfolio loss (in million) with indicators for EL, VaR and ES
alpha=c(0.995,0.999)
plot(TestModel,1e6,alpha=alpha)

#calculate portfolio VaR and ES
VaR=VaR(TestModel,alpha)
ES=ES(TestModel,alpha)

#Calculate risk contributions to VaR and ES
risk.cont=cbind(VaR.cont(TestModel,alpha = alpha),
ES.cont(TestModel,alpha = alpha))

```

**Description**

Get the values of LGD, defined within the portfolio

**Usage**

```
LGD(this)
```

**Arguments**

`this`            Object of class GCPM

**Value**

numeric vector of length equal to number of counterparties

**See Also**

[portfolio.pois](#)

---

LHR-methods

*Likelihood Ratio*

---

**Description**

Get the likelihood ratio for each scenario defined in `random.numbers` (see [init](#))

**Usage**

```
LHR(this)
```

**Arguments**

`this`            Object of class GCPM

**Value**

numeric vector of length equal to `nrow(random.numbers)`

---

link.function-methods *Model Link Function*

---

**Description**

Get the models link function (see [init](#))

**Usage**

```
link.function(this)
```

**Arguments**

this            Object of class GCPM

**Value**

character value of length 1

**See Also**

[init](#)

---

loss-methods            *Loss Levels*

---

**Description**

Get the loss levels of the portfolio loss distribution.

**Usage**

```
loss(this)
```

**Arguments**

this            Object of class GCPM

**Value**

numeric vector

---

loss.thr-methods      *Threshold of Saved Portfolio Loss*

---

**Description**

Get the value of loss.thr (see [init](#))

**Usage**

```
loss.thr(this)
```

**Arguments**

this                  Object of class GCPM

**Value**

numeric value of length 1

**See Also**

[init](#)

---

loss.unit-methods      *Loss Unit*

---

**Description**

Get the loss unit used for potential loss discretization of the model

**Usage**

```
loss.unit(this)
```

**Arguments**

this                  Object of class GCPM

**Value**

numeric value of length 1

**See Also**

[init](#)

---

model.type-methods	<i>Model Type</i>
--------------------	-------------------

---

**Description**

Get the value of model.type (see [init](#))

**Usage**

```
model.type(this)
```

**Arguments**

this	Object of class GCPM
------	----------------------

**Value**

character value of length 1

**See Also**

[init](#)

---

N-methods	<i>Number of Simulations</i>
-----------	------------------------------

---

**Description**

Get the value of N (number of simulations, see [init](#))

**Usage**

```
N(this)
```

**Arguments**

this	Object of class GCPM
------	----------------------

**Value**

numeric value of length 1

**See Also**

[init](#)

---

name-methods

*Counterparty Names*

---

**Description**

Get the value of name, i.e. the counterparties' names, defined in the portfolio (see [analyze](#))

**Usage**

```
name(this)
```

**Arguments**

this            Object of class GCPM

**Value**

character value of length equal to number of counterparties

**See Also**

[portfolio.pois](#)

---

NC-methods

*Number of Counterparties*

---

**Description**

Get the value of NC, representing the number of counterparties within the portfolio (see [analyze](#))

**Usage**

```
NC(this)
```

**Arguments**

this            Object of class GCPM

**Value**

numeric value of length 1

**See Also**

[analyze](#)



---

NR-methods

*Counterparty IDs*

---

**Description**

Get the value of NR, the counterparties' identification numbers within the portfolio (see [analyze](#))

**Usage**

NR(this)

**Arguments**

this            Object of class GCPM

**Value**

numeric value of length equal to number of counterparties

**See Also**

[portfolio.pois](#)

---

NS-methods

*Number of Sectors*

---

**Description**

Get the value of NS, the number of sectors within the model (see [init](#))

**Usage**

NS(this)

**Arguments**

this            Object of class GCPM

**Value**

numeric value of length 1

**See Also**

[init](#)

---

PD-methods

*Counterparty Probability of Default*

---

**Description**

Get the value of PD, the counterparties default probabilities within the portfolio (see [analyze](#). Please note, that these PDs are adjusted because of discretization in order to preserve the expected loss.)

**Usage**

PD(this)

**Arguments**

this            Object of class GCPM

**Value**

numeric value of length equal to the number of counterparties

**See Also**

[portfolio.pois](#)

---

PDF-methods

*Probability Density Function*

---

**Description**

Get the value of PDF, representing the pdf of the estimated portfolio loss distribution.

**Usage**

PDF(this)

**Arguments**

this            Object of class GCPM

**Value**

numeric vector

---

 PL-methods

*Counterparty Potential Loss*


---

**Description**

Get the value of PL, the potential losses of counterparties (see [GCPM-class](#)). Please note, that the potential losses are discretized according to `loss.unit` (see [init](#)).

**Usage**

```
PL(this)
```

**Arguments**

<code>this</code>	Object of class GCPM
-------------------	----------------------

**Value**

numeric value of length equal to the number of counterparties

**See Also**

[portfolio.pois](#), [init](#)

---

 plot-methods

*Plot of the Portfolio Loss Distribution*


---

**Description**

Plot of the estimated pdf of the portfolio loss distribution.

**Usage**

```
plot(x,y,...)
```

**Arguments**

<code>x</code>	Object of class GCPM
<code>y</code>	plot unit for losses (x-axis), default value = 1
<code>...</code>	Further arguments such as: alpha If provided vertical lines are added, representing value at risk and expected shortfall on level(s) alpha or nbins number of supporting points, default value = 100

---

`portfolio.pois`*Example Portfolio Data with Poisson Default Mode*

---

**Description**

The dataset contains an example portfolio in the structure needed by the [analyze](#) function.

**Usage**

```
data("portfolio.pois")
```

**Format**

A data frame with 3000 counterparties and the following variables.

Number Counterparty ID (numeric)

Name Counterparty name (character)

Business Business line (character)

Country Country (character)

EAD Exposure at default (numeric)

LGD Loss given default (numeric)

PD Probability of default (numeric)

Default Default mode ('Poisson' or 'Benroulli')

A sector weights for sector A

B sector weights for sector B

C sector weights for sector C

---

`portfolio.pool`*Pooled Portfolio*

---

**Description**

In order to speed up calculations, counterparties of [portfolio.pois](#) with  $EAD * LGD < 200,000$  are grouped together (pooled).

**Usage**

```
data("portfolio.pool")
```

**Format**

A data frame with 1400 counterparties and 3 pools (each per sector) and the following variables.

Number Counterparty ID (numeric)

Name Counterparty name (character)

Business Business line (character)

Country Country (character)

EAD Exposure at default (numeric); pool: average EAD per counterparty

LGD Loss given default (numeric); pool: EAD-weighted average LGD per counterparty

PD Probability of default (numeric); pool: expectation of number of defaults

Default Default mode ('Poisson' for pools or 'Benroulli')

A sector weights for sector A

B sector weights for sector B

C sector weights for sector C

---

portfolios

*Example Portfolios for GCPM Package*

---

**Description**

The workspace contain the example portfolio (with Poisson default mode) in the structure needed by the [analyze](#) function as well as a pooled version.

**Usage**

```
data("portfolios")
```

**Format**

Two data frames containing the portfolios.

**See Also**

[portfolio.pois](#), [portfolio.pool](#), [analyze](#)

---

random.numbers-methods

*Sector Drawings*

---

**Description**

Get the content of `random.numbers`, representing the sector drawings (see [init](#))

**Usage**

```
random.numbers(this)
```

**Arguments**

`this`            Object of class GCPM

**Value**

numeric matrix

**See Also**

[init](#)

---

SD-methods

*Standard Deviation (Loss Distribution)*

---

**Description**

Get the value of SD, the portfolio standard deviation derived from the loss distribution.

**Usage**

```
SD(this)
```

**Arguments**

`this`            Object of class GCPM

**Value**

numeric value of length 1

---

SD.analyt-methods	<i>Standard Deviation (from Portfolio Data)</i>
-------------------	---

---

**Description**

Get the value of SD.analyt, the portfolio standard deviation derived from the portfolio data (see [GCPM-class](#)). This value is only available in case of an analytical model.

**Usage**

```
SD.analyt(this)
```

**Arguments**

this	Object of class GCPM
------	----------------------

**Value**

numeric value of length 1

---

SD.cont-methods	<i>Risk Contributions to Portfolio Standard Deviation</i>
-----------------	---

---

**Description**

Get the counterparties' contributions to portfolio standard deviation (see [GCPM-class](#)). These values are only available in case of an analytical model.

**Usage**

```
SD.cont(this)
```

**Arguments**

this	Object of class GCPM
------	----------------------

**Value**

numeric value of length equal to number of counterparties

---

SD.div-methods	<i>Diversifiable Risk (Standard Deviation)</i>
----------------	--

---

**Description**

Get the value of SD.div, the diversifiable part of portfolio standard deviation (see [GCPM-class](#))

**Usage**

```
SD.div(this)
```

**Arguments**

this	Object of class GCPM
------	----------------------

**Value**

numeric value of length 1

---

SD.syst-methods	<i>Systemic Risk (Standard Deviation)</i>
-----------------	---

---

**Description**

Get the value of SD.syst, the non-diversifiable part of portfolio standard deviation.

**Usage**

```
SD.syst(this)
```

**Arguments**

this	Object of class GCPM
------	----------------------

**Value**

numeric value of length 1



---

sec.var-methods	<i>Sector Variances</i>
-----------------	-------------------------

---

**Description**

Get the value of `sec.var`, the sector variances in case of an analytical CreditRisk+ like model (see [init](#))

**Usage**

```
sec.var(this)
```

**Arguments**

<code>this</code>	Object of class GCPM
-------------------	----------------------

**Value**

numeric value of length equal to number of sectors

**See Also**

[init](#)

---

sector.names-methods	<i>Sector Names</i>
----------------------	---------------------

---

**Description**

Get the value of `sector.names`, the sector names (see [init](#))

**Usage**

```
sector.names(this)
```

**Arguments**

<code>this</code>	Object of class GCPM
-------------------	----------------------

**Value**

factor of length equal to number of sectors

**See Also**

[init](#)

---

seed-methods	<i>Random Number Seed</i>
--------------	---------------------------

---

**Description**

Get the value of seed (see [init](#))

**Usage**

```
seed(this)
```

**Arguments**

this	Object of class GCPM
------	----------------------

**Value**

numeric value of length 1

**See Also**

[init](#)

---

show-methods	<i>Show Parameters of Credit Portfolio Model</i>
--------------	--

---

**Description**

Displays the most important parameters and portfolio statistics (if available).

---

summary-methods	<i>Model summary</i>
-----------------	----------------------

---

**Description**

Create a Summary List with Model Parameters.

**Usage**

```
summary(object,...)
```

**Arguments**

object	Object of class GCPM
...	No further arguments

**Value**

list

---

`VaR-methods`*Portfolio Value at Risk*

---

**Description**

Calculate the portfolio value at risk on level(s) alpha.

**Usage**`VaR(this, alpha)`**Arguments**

<code>this</code>	Object of class GCPM
<code>alpha</code>	numeric vector with entries between 0 and 1

**Value**

numeric value of length equal to length of alpha

---

`VaR.cont-methods`*Risk Contributions to Portfolio Value at Risk*

---

**Description**

Get the counterparties' contributions to portfolio value at risk (see [GCPM-class](#)). In case of a simulative model, these values are calculated from individual losses greater or equal `loss.thr` (see [init](#)). Contributions are not available if `loss.thr` is too high.

**Usage**`VaR.cont(this, alpha)`**Arguments**

<code>this</code>	Object of class GCPM
<code>alpha</code>	numeric vector with entries between 0 and 1

**Value**

numeric matrix

**See Also**

[init](#), [loss.thr](#)

---

W-methods

*Sector Weights*

---

**Description**

Get the value of W, the matrix of counterparties' sector weights defined within the portfolio (see [analyze](#))

**Usage**

W(this)

**Arguments**

this            Object of class GCPM

**Value**

numeric matrix

**See Also**

[init](#)

# Index

- \* **GCPM**
  - analyze-methods, 5
- \* **classes**
  - GCPM-class, 14
- \* **datasets**
  - portfolio.pois, 28
  - portfolio.pool, 28
  - portfolios, 29
- \* **methods**
  - alpha.max-methods, 4
  - analyze-methods, 5
  - business-methods, 7
  - CDF-methods, 8
  - country-methods, 9
  - default-methods, 9
  - EAD-methods, 10
  - EC-methods, 10
  - EC.cont-methods, 11
  - EL-methods, 11
  - EL.analyt-methods, 12
  - ES-methods, 13
  - ES.cont-methods, 13
  - export-methods, 14
  - idiosyncr-methods, 17
  - LGD-methods, 19
  - LHR-methods, 20
  - link.function-methods, 21
  - loss-methods, 21
  - loss.thr-methods, 22
  - loss.unit-methods, 22
  - model.type-methods, 23
  - N-methods, 23
  - name-methods, 24
  - NC-methods, 24
  - NR-methods, 25
  - NS-methods, 25
  - PD-methods, 26
  - PDF-methods, 26
  - PL-methods, 27
  - plot-methods, 27
  - random.numbers-methods, 30
  - SD-methods, 30
  - SD.analyt-methods, 31
  - SD.cont-methods, 31
  - SD.div-methods, 32
  - SD.syst-methods, 32
  - sec.var-methods, 33
  - sector.names-methods, 33
  - seed-methods, 34
  - summary-methods, 34
  - VaR-methods, 35
  - VaR.cont-methods, 35
  - W-methods, 36
- \* **package**
  - GCPM-package, 3
  - alpha.max (alpha.max-methods), 4
  - alpha.max, GCPM (alpha.max-methods), 4
  - alpha.max-methods, 4
  - analyze, 3, 8, 16, 18, 24–26, 28, 29, 36
  - analyze (analyze-methods), 5
  - analyze, GCPM, data.frame, missing, missing-method (analyze-methods), 5
  - analyze, GCPM, data.frame, missing, numeric-method (analyze-methods), 5
  - analyze, GCPM, data.frame, numeric, missing-method (analyze-methods), 5
  - analyze, GCPM, data.frame, numeric, numeric-method (analyze-methods), 5
  - analyze, GCPM-method (analyze-methods), 5
  - analyze-methods, 5
  - business (business-methods), 7
  - business, GCPM-method (business-methods), 7
  - business-methods, 7
  - CDF (CDF-methods), 8
  - CDF, GCPM-method (CDF-methods), 8

- CDF-methods, [8](#)
- country (country-methods), [9](#)
- country, GCPM-method (country-methods), [9](#)
- country-methods, [9](#)
  
- default (default-methods), [9](#)
- default, GCPM-method (default-methods), [9](#)
- default-methods, [9](#)
  
- EAD (EAD-methods), [10](#)
- EAD, GCPM-method (EAD-methods), [10](#)
- EAD-methods, [10](#)
- EC, [6](#)
- EC (EC-methods), [10](#)
- EC, GCPM, missing-method (EC-methods), [10](#)
- EC, GCPM, numeric-method (EC-methods), [10](#)
- EC-methods, [10](#)
- EC.cont (EC.cont-methods), [11](#)
- EC.cont, GCPM, numeric-method (EC.cont-methods), [11](#)
- EC.cont, GCPM-method (EC.cont-methods), [11](#)
- EC.cont-methods, [11](#)
- EL, [12](#)
- EL (EL-methods), [11](#)
- EL, GCPM-method (EL-methods), [11](#)
- EL-methods, [11](#)
- EL.analyt, [11](#), [12](#)
- EL.analyt (EL.analyt-methods), [12](#)
- EL.analyt, GCPM-method (EL.analyt-methods), [12](#)
- EL.analyt-methods, [12](#)
- ES, [6](#)
- ES (ES-methods), [13](#)
- ES, GCPM, missing-method (ES-methods), [13](#)
- ES, GCPM, numeric-method (ES-methods), [13](#)
- ES-methods, [13](#)
- ES.cont (ES.cont-methods), [13](#)
- ES.cont, GCPM, numeric-method (ES.cont-methods), [13](#)
- ES.cont, GCPM-method (ES.cont-methods), [13](#)
- ES.cont-methods, [13](#)
- export (export-methods), [14](#)
- export, GCPM, character, character, missing-method (export-methods), [14](#)
- export, GCPM, character, character, numeric-method (export-methods), [14](#)
- export, GCPM, character, missing, missing-method (export-methods), [14](#)
- export, GCPM, character, missing, numeric-method (export-methods), [14](#)
- export, GCPM, missing, character, missing-method (export-methods), [14](#)
- export, GCPM, missing, character, numeric-method (export-methods), [14](#)
- export, GCPM, missing, missing, missing-method (export-methods), [14](#)
- export, GCPM, missing, missing, numeric-method (export-methods), [14](#)
- export, GCPM-method (export-methods), [14](#)
- export-methods, [14](#)
  
- GCPM, [18](#)
- GCPM (GCPM-package), [3](#)
- GCPM-class, [14](#)
- GCPM-package, [3](#)
  
- idiosyncr (idiosyncr-methods), [17](#)
- idiosyncr, GCPM-method (idiosyncr-methods), [17](#)
- idiosyncr-methods, [17](#)
- init, [3](#), [5](#), [6](#), [14](#), [16](#), [17](#), [20–23](#), [25](#), [27](#), [30](#), [33–36](#)
  
- LGD (LGD-methods), [19](#)
- LGD, GCPM-method (LGD-methods), [19](#)
- LGD-methods, [19](#)
- LHR (LHR-methods), [20](#)
- LHR, GCPM-method (LHR-methods), [20](#)
- LHR-methods, [20](#)
- link.function (link.function-methods), [21](#)
- link.function, GCPM-method (link.function-methods), [21](#)
- link.function-methods, [21](#)
- loss (loss-methods), [21](#)
- loss, GCPM-method (loss-methods), [21](#)
- loss-methods, [21](#)
- loss.thr, [11](#), [13](#), [35](#)
- loss.thr (loss.thr-methods), [22](#)
- loss.thr, GCPM-method (loss.thr-methods), [22](#)
- loss.thr-methods, [22](#)
- loss.unit (loss.unit-methods), [22](#)
- loss.unit, GCPM-method (loss.unit-methods), [22](#)

- loss.unit-methods, 22
- model.type (model.type-methods), 23
- model.type, GCPM-method  
(model.type-methods), 23
- model.type-methods, 23
- N (N-methods), 23
- N, GCPM-method (N-methods), 23
- N-methods, 23
- name (name-methods), 24
- name, GCPM-method (name-methods), 24
- name-methods, 24
- NC (NC-methods), 24
- NC, GCPM-method (NC-methods), 24
- NC-methods, 24
- NR (NR-methods), 25
- NR, GCPM-method (NR-methods), 25
- NR-methods, 25
- NS (NS-methods), 25
- NS, GCPM-method (NS-methods), 25
- NS-methods, 25
- PD (PD-methods), 26
- PD, GCPM-method (PD-methods), 26
- PD-methods, 26
- PDF (PDF-methods), 26
- PDF, GCPM-method (PDF-methods), 26
- PDF-methods, 26
- PL (PL-methods), 27
- PL, GCPM-method (PL-methods), 27
- PL-methods, 27
- plot (plot-methods), 27
- plot, ANY-method (plot-methods), 27
- plot, GCPM-method (plot-methods), 27
- plot-methods, 27
- portfolio.pois, 8–10, 20, 24–28, 28, 29
- portfolio.pool, 28, 29
- portfolios, 29
- random.numbers  
(random.numbers-methods), 30
- random.numbers, GCPM-method  
(random.numbers-methods), 30
- random.numbers-methods, 30
- SD (SD-methods), 30
- SD, GCPM-method (SD-methods), 30
- SD-methods, 30
- SD.analyt (SD.analyt-methods), 31
- SD.analyt, GCPM-method  
(SD.analyt-methods), 31
- SD.analyt-methods, 31
- SD.cont (SD.cont-methods), 31
- SD.cont, GCPM-method (SD.cont-methods),  
31
- SD.cont-methods, 31
- SD.div (SD.div-methods), 32
- SD.div, GCPM-method (SD.div-methods), 32
- SD.div-methods, 32
- SD.syst (SD.syst-methods), 32
- SD.syst, GCPM-method (SD.syst-methods),  
32
- SD.syst-methods, 32
- sec.var (sec.var-methods), 33
- sec.var, GCPM-method (sec.var-methods),  
33
- sec.var-methods, 33
- sector.names (sector.names-methods), 33
- sector.names, GCPM-method  
(sector.names-methods), 33
- sector.names-methods, 33
- seed (seed-methods), 34
- seed, GCPM-method (seed-methods), 34
- seed-methods, 34
- show, GCPM-method (show-methods), 34
- show-methods, 34
- summary (summary-methods), 34
- summary, ANY-method (summary-methods), 34
- summary, GCPM-method (summary-methods),  
34
- summary-methods, 34
- VaR, 6
- VaR (VaR-methods), 35
- VaR, GCPM, missing-method (VaR-methods),  
35
- VaR, GCPM, numeric-method (VaR-methods),  
35
- VaR, GCPM-method (VaR-methods), 35
- VaR-methods, 35
- VaR.cont (VaR.cont-methods), 35
- VaR.cont, GCPM, numeric-method  
(VaR.cont-methods), 35
- VaR.cont, GCPM-method  
(VaR.cont-methods), 35
- VaR.cont-methods, 35

W (W-methods), [36](#)

W, GCPM-method (W-methods), [36](#)

W-methods, [36](#)