

# Package ‘Trading’

April 13, 2025

**Type** Package

**Title** Trade Objects, Advanced Correlation & Beta Estimates, Betting Strategies

**Version** 3.2

**Date** 2025-04-13

**Author** Tasos Grivas [aut, cre]

**Maintainer** Tasos Grivas <info@openriskcalculator.com>

**Description** Contains performance analysis metrics of track records including entropy-based correlation and dynamic beta based on a state/space algorithm. The normalized sample entropy method has been implemented which produces accurate entropy estimation even on smaller datasets. On a separate stream, trades from the five major assets classes and also functionality to use pricing curves, rating tables, Credit Support Annex and add-on tables. The implementation follows an object oriented logic whereby each trade inherits from more abstract classes while also the curves/tables are objects. Furthermore, odds calculators and P&L back-testing functionality has been implemented for the most widely used betting/trading strategies including martingale, 'DAlembert', 'Labouchere' and Fibonacci. Back testing has also been included for the 'EuroMillions', the 'EuroJackpot', the UK Lotto, the Set For Life and the UK 'ThunderBall' lotteries. Furthermore, some basic functionality about climate risk has been included.

**Imports** methods, reticulate, PerformanceAnalytics, data.table, ggplot2, readxl, RcppAlgos

**URL** <https://openriskcalculator.com/>

**License** GPL-3

**Collate** 'AngularDistance.R' 'Future.R' 'Swap.R' 'Vol.R' 'Option.R' 'Trade.R' 'IRD.R' 'Bond.R' 'CSA.R' 'CalcEuroLotteryPnL.R' 'CalcSetForLifePnL.R' 'CalcUKLotteryPnL.R' 'CalcUKThunderBallPnL.R' 'Chebyshev\_distance.R' 'Collateral.R' 'Commodity.R' 'Credit.R' 'CrossSampleEntropy.R' 'Curve.R' 'DynamicBeta.R' 'Equity.R' 'EuroJackpotResults.R' 'EuroLotteryAllCombinations.R' 'EuroLotteryBacktesting.R' 'EuroMillionsResults.R' 'FX.R' 'GetTradeDetails.R'

'HashTable.R' 'InformationAdjustedBeta.R'  
 'InformationAdjustedCorr.R' 'NormXASampEn.R' 'Other.R'  
 'OuterJoinMerge.R' 'ParseTrades.R' 'SampleEntropy.R'  
 'SelectDerivatives.R' 'SetForLifeBacktesting.R'  
 'SetForLifeExample.R' 'SetForLifeResults.R'  
 'UKLotteryBacktesting.R' 'UKLotteryExample.R'  
 'UKLotteryResults.R' 'UKThunderBallBacktesting.R'  
 'UKThunderBallExample.R' 'UKThunderBallResults.R'  
 'VariationOfInformation.R' 'capped\_fibonacci\_seq.R' 'cf.R'  
 'ci.R' 'eurojackpotExample.R' 'euromillionsExample.R'  
 'globals.R' 'martingale\_strategy\_calculator.R' 'onLoad.R'  
 'roulette\_pl\_calculator\_Labouchere.R'  
 'roulette\_pl\_calculator\_dalembert.R'  
 'roulette\_pl\_calculator\_fibonacci.R'  
 'roulette\_pl\_calculator\_martingale.R'  
 'roulette\_pl\_calculator\_specific\_number.R' 'tce.R' 'top5.R'  
 'waci.R'

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2025-04-13 12:50:02 UTC

## Contents

AngularDistance . . . . .	4
Bond-class . . . . .	5
BondFuture-class . . . . .	6
CalcEuroLotteryPnL . . . . .	7
CalcSetForLifePnL . . . . .	7
CalcUKLotteryPnL . . . . .	8
CalcUKThunderBallPnL . . . . .	9
capped_fibonacci_seq . . . . .	10
Carbon_Footprint . . . . .	10
Carbon_Intensity . . . . .	11
CDOTranche-class . . . . .	12
CDS-class . . . . .	13
CDX-class . . . . .	14
Chebyshev_distance . . . . .	14
Collateral-class . . . . .	15
Commodity-class . . . . .	16
CommodityForward-class . . . . .	17
CommSwap-class . . . . .	18
CrossSampleEntropy . . . . .	18
CSA-class . . . . .	19
Curve-class . . . . .	20
DynamicBeta . . . . .	21

Equity-class . . . . . 22

EquityIndexFuture-class . . . . . 22

EquityOptionIndex-class . . . . . 23

EquityOptionSingle-class . . . . . 24

EuroJackpotExample . . . . . 24

EuroJackpotResults . . . . . 25

EuroLotteryAllCombinations . . . . . 25

EuroLotteryBacktesting . . . . . 26

EuroMillionsExample . . . . . 27

EuroMillionsResults . . . . . 27

FxForward-class . . . . . 28

FxSwap-class . . . . . 29

GetTradeDetails . . . . . 30

HashTable-class . . . . . 30

InformationAdjustedBeta . . . . . 31

InformationAdjustedCorr . . . . . 32

IRDFuture-class . . . . . 33

IRDSwap-class . . . . . 33

IRDSwaption-class . . . . . 34

IRDSwapVol-class . . . . . 35

martingale\_strategy\_repetitions . . . . . 35

NormXASampEn . . . . . 36

OtherExposure-class . . . . . 37

OuterJoinMerge . . . . . 38

ParseTrades . . . . . 39

roulette\_pl\_calculator\_dalembert . . . . . 39

roulette\_pl\_calculator\_fibonacci . . . . . 41

roulette\_pl\_calculator\_labouchere . . . . . 42

roulette\_pl\_calculator\_martingale . . . . . 43

roulette\_pl\_calculator\_specific\_number . . . . . 44

SampleEntropy . . . . . 46

SelectDerivatives . . . . . 47

SetForLifeBacktesting . . . . . 47

SetForLifeExample . . . . . 48

SetForLifeResults . . . . . 49

top5 . . . . . 49

Total\_Carbon\_Emissions . . . . . 50

UKLotteryBacktesting . . . . . 51

UKLotteryExample . . . . . 52

UKLotteryResults . . . . . 52

UKThunderballBacktesting . . . . . 53

UKThunderballExample . . . . . 53

UKThunderBallResults . . . . . 54

VariationOfInformation . . . . . 55

Weighted\_Average\_Carbon\_Intensity . . . . . 56

---

AngularDistance	<i>Angular distance metrics</i>
-----------------	---------------------------------

---

**Description**

Calculates the angular distance between a matrix of the track records of various assets/strategies. The sign of the correlation can be ignored for long/short portfolios.

**Usage**

```
AngularDistance(returns_matrix, long_short = FALSE)
```

**Arguments**

`returns_matrix` a matrix containing the track records of the underlying assets/strategies.  
`long_short` a boolean value which results in the sign of the correlation being ignored, default value is FALSE

**Value**

A matrix containing the angular distance values.

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

Lopez de Prado, Marcos, Codependence (Presentation Slides) (January 2, 2020). Available at SSRN: <https://ssrn.com/abstract=3512994>

**Examples**

```
## calling AngularDistance() without an argument loads the historical edhec data
## for the "Short Selling" and "Convertible Arbitrage" strategies
returns_matrix = PerformanceAnalytics::edhec[,c("Short Selling", "Convertible Arbitrage")]
angular_distance = AngularDistance(returns_matrix, long_short=FALSE)
```

Bond-class

*Bond Class***Description**

Creates a Bond object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
BuySell	Takes the values of either 'Buy' or 'Sell'
yield	The yield of the Bond
ISIN	The ISIN of the Bond,
payment_frequency	the frequency that the bond pays coupon (Quarter, SA etc)
maturity_date	the maturity date of the bond
coupon_type	The coupon type of the bond (fixed, floating, flipper etc)
credit_risk_weight	The percentage weight of the exposure of the bond that should be attributed to the 'Credit' asset class
Issuer	The issuer of the bond

**Value**

An object of type Bond

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
tr1 = Bond(Notional=10000,MtM=30,Currency="EUR",Si=0,maturity_date="2026-04-04",
BuySell='Buy',payment_frequency="SA",
credit_risk_weight=0.2,coupon_type="Fixed",Issuer="FirmA",ISIN = "XS0943423")
```

---

BondFuture-class      *Bond Future Class*

---

### Description

Creates a Bond Future object with the relevant info needed to calculate the Exposure-at-Default (EAD)

### Arguments

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
yield	The yield of the Underlying Bond
isin	The ISIN of the Underlying Bond,
payment_frequency	the frequency that the bond pays coupon (Quarter, SA etc)
maturity_date	the maturity date of the bond
coupon_type	The coupon type of the bond (fixed, floating, flipper etc)
Issuer	The issuer of the bond

### Value

An object of type Bond

### Author(s)

Tasos Grivas <tasos@openriskcalculator.com>

### Examples

```
example_trades = ParseTrades()
bondfuture_trade = example_trades[[17]]
tr1 = BondFuture(Notional=10000, MtM=30, Currency="EUR", Si=0, Ei=10, BuySell='Buy',
payment_frequency="SA", coupon_type="Fixed", Issuer="CountryA", ISIN = "XS0943423")
```

---

CalcEuroLotteryPnL      *PnL calculation for EuroMillions/EuroJackpot backtesting*

---

**Description**

Calculates the PnL for a pay out structure created during backtesting

**Usage**

```
CalcEuroLotteryPnL(backtested_results, plot_results = FALSE)
```

**Arguments**

`backtested_results`      The EuroMillions/EuroJackpot results backtested against the user input  
`plot_results`      (Optional) If TRUE, the P&L historical graphs are plotted, default FALSE

**Value**

PnL figures

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
euromillions_results = EuroMillionsResults()  
user_input = c(10,20,30,40,50,5,10)  
backtested_results = EuroLotteryBacktesting(euromillions_results, '2005-01-01', user_input)  
pnl_result = CalcEuroLotteryPnL(backtested_results, plot_results = TRUE)
```

---

CalcSetForLifePnL      *PnL calculation for Set For Life backtesting*

---

**Description**

Calculates the PnL for a pay out structure created during backtesting

**Usage**

```
CalcSetForLifePnL(backtested_results, plot_results = FALSE)
```

**Arguments**

backtested\_results      The Set For Life results backtested against the user input  
plot\_results      (Optional) If TRUE, the P&L historical graphs are plotted, default FALSE

**Value**

PnL figures

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
setforlife_results = SetForLifeResults()  
user_input = c(10,20,30,40,50,5)  
backtested_results = SetForLifeBacktesting(setforlife_results, date_since='2005-01-01', user_input)  
pnl_result = CalcSetForLifePnL(backtested_results, plot_results = FALSE)
```

---

CalcUKLotteryPnL      *PnL calculation for UKLottery backtesting*

---

**Description**

Calculates the PnL for a pay out structure created during backtesting

**Usage**

```
CalcUKLotteryPnL(backtested_results, plot_results = FALSE)
```

**Arguments**

backtested\_results      The UKLottery results backtested against the user input  
plot\_results      (Optional) If TRUE, the P&L historical graphs are plotted, default FALSE

**Value**

PnL figures

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>



**Examples**

```
uklottery_results = UKLotteryResults()
user_input = c(5,10,20,30,40,50)
backtested_results = UKLotteryBacktesting(uklottery_results,, user_input)
pnl_result = CalcUKLotteryPnL(backtested_results, plot_results = FALSE)
```

---

CalcUKThunderBallPnL *PnL calculation for UKThunderBall backtesting*

---

**Description**

Calculates the PnL for a pay out structure created during backtesting

**Usage**

```
CalcUKThunderBallPnL(backtested_results, plot_results = FALSE)
```

**Arguments**

backtested\_results  
The UKThunderBall results backtested against the user input

plot\_results (Optional) If TRUE, the P&L historical graphs are plotted, default FALSE

**Value**

PnL figures

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
ukthunderball_results = UKThunderBallResults()
user_input = c(10,20,30,31,32,5)
backtested_results = UKThunderballBacktesting(ukthunderball_results, user_input = user_input)
pnl_result = CalcUKThunderBallPnL(backtested_results, plot_results = FALSE)
```

---

capped\_fibonacci\_seq    *Fibonacci sequence up to a specified maximum number*

---

**Description**

Generates the Fibonacci sequence up to a specified maximum number

**Usage**

```
capped_fibonacci_seq(max_number)
```

**Arguments**

max\_number    The maximum number up to which the sequence should be generated

**Value**

A vector containing the Fibonacci sequence

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

[https://en.wikipedia.org/wiki/Fibonacci\\_number](https://en.wikipedia.org/wiki/Fibonacci_number)

**Examples**

```
fibonacci_seq = capped_fibonacci_seq(max_number = 6000)
```

---

Carbon\_Footprint    *Carbon Footprint*

---

**Description**

Returns the Total carbon emissions for a portfolio normalized by the market value of the portfolio, expressed in tons CO<sub>2</sub>e / \$M invested. Scope 1 and Scope 2 GHG emissions are allocated to investors based on an equity

**Usage**

```
Carbon_Footprint(portfolio_exposure, emissions_capitalization_data)
```

**Arguments**

portfolio\_exposure  
The exposure per issuer in the portfolio

emissions\_capitalization\_data  
The capitalization and the Scope 1 & 2 GHG emissions per issuer

**Value**

Total carbon emissions for a portfolio normalized by the market value of the portfolio, expressed in tons CO<sub>2</sub>e / \$M invested.

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

<https://www.tcfhub.org/Downloads/pdfs/E09>

**Examples**

```
portfolio_exposure = data.table::data.table(Issuers = c('A','B','C'),
exposures = c(100, 200, 50))
emissions_capitalization_data = data.table::data.table(Issuers = c('A','B','C'),
emissions = c(1000, 5000, 6000), Capitalization = c(20000, 10000, 30000))
Carbon_Footprint(portfolio_exposure, emissions_capitalization_data)
```

---

Carbon_Intensity	<i>Carbon Intensity</i>
------------------	-------------------------

---

**Description**

Returns the Volume of carbon emissions per million dollars of revenue expressed in tons CO<sub>2</sub>e / \$M revenue. Scope 1 and Scope 2 GHG emissions are allocated to investors based on an equity ownership approach. The company's (or issuer's) revenue is used to adjust for company size to provide a measurement of the efficiency of output.

**Usage**

```
Carbon_Intensity(portfolio_exposure, emissions_capitalization_revenue_data)
```

**Arguments**

portfolio\_exposure  
The exposure per issuer in the portfolio

emissions\_capitalization\_revenue\_data  
The capitalization, revenue and the Scope 1 & 2 GHG emissions per issuer

**Value**

Volume of carbon emissions per million dollars of revenue expressed in tons CO<sub>2</sub>e / \$M revenue.

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

<https://www.tcfhub.org/Downloads/pdfs/E09>

**Examples**

```
portfolio_exposure      = data.table::data.table(Issuers = c('A','B','C'),
  exposures = c(100, 200, 50))
emissions_capitalization_revenue_data = data.table::data.table(Issuers = c('A','B','C'),
  emissions = c(1000, 5000, 6000), revenue = c(2000, 5000, 3000),Capitalization =
  c(20000, 10000, 15000))
Carbon_Intensity (portfolio_exposure, emissions_capitalization_revenue_data)
```

---

CDOTranche-class

*CDO tranche Class*

---

**Description**

Creates a CDO tranche Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the belongs
Si	The number of years after which the trade will start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
attach_point	The attachment point of the tranche
detach_point	The detachment point of the tranche

**Value**

An object of type CDOTranche

**Examples**

```
## a CDO tranche object
tr3 = CDOTranche(Notional=10000,MtM=0,Currency="USD",Si=0,Ei=5,
BuySell='Buy',SubClass='IG',RefEntity='CDX.IG',cdo_attach_point=0.3 ,cdo_detach_point=0.5)
```

---

CDS-class

*CDS Class*


---

### Description

Creates a CDS Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

### Arguments

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
SubClass	Specifies the rating of the underlying entity (possible values are A, AA, BB etc)
RefEntity	The name of the underlying entity

### Value

An object of type CDS

### Author(s)

Tasos Grivas <tasos@openriskcalculator.com>

### References

Basel Committee: The standardised approach for measuring counterparty credit risk exposures  
<http://www.bis.org/publ/bcbs279.htm>

### Examples

```
## the CDS trade given in the Basel regulation Credit example
tr1 = CDS(Notional=10000,MtM=20,Currency="USD",Si=0,Ei=3,BuySell='Buy',
SubClass='AA',RefEntity='FirmA')
```

---

CDX-class	<i>CDX Class</i>
-----------	------------------

---

**Description**

Creates a Credit Index Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the belongs
Si	The number of years after which the trade will start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
SubClass	Specifies if the underlying Index is investment grade or not (possible values are IG & SG)
RefEntity	The name of the underlying Index

**Value**

An object of type CDX

**Examples**

```
## the CDX trade given in the Basel regulation Credit example
tr3 = CDX(Notional=10000,MtM=0,Currency="USD",Si=0,Ei=5,
BuySell='Buy',SubClass='IG',RefEntity='Portfolio_1')
```

---

Chebyshev_distance	<i>Chebyshev distance</i>
--------------------	---------------------------

---

**Description**

Calculates the Chebyshev distance

**Usage**

Chebyshev\_distance(x, y)

**Arguments**

x	a vector containing the track record of the underlying asset/strategy
y	a vector containing the track record of the underlying asset/strategy

**Value**

The Chebyshev distance of the two vectors

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

[https://en.wikipedia.org/wiki/Chebyshev\\_distance](https://en.wikipedia.org/wiki/Chebyshev_distance)

**Examples**

```
x = rnorm(1000)
y = rnorm(1000)

chebyshev_dist = Chebyshev_distance(x, y)
```

---

Collateral-class

*Collateral Class*

---

**Description**

Creates a Collateral amount object which needs to be linked with a CSA ID

**Arguments**

ID	The ID of each object
Amount	The collateral amount
csa_id	The csa_id that this object is linked with
type	Describes the type of the collateral: can be "ICA", "VariationMargin" etc

**Value**

An object of type Collateral

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

Basel Committee: The standardised approach for measuring counterparty credit risk exposures  
<http://www.bis.org/publ/bcbs279.htm>

**Examples**

```

colls = list()
coll_raw = read.csv(system.file("extdata", "coll.csv", package = "Trading"),header=TRUE,
stringsAsFactors = FALSE)

for(i in 1:nrow(coll_raw))
{
  colls[[i]] = Collateral()
  colls[[i]]$PopulateViaCSV(coll_raw[i,])
}

```

Commodity-class

*Commodity Class***Description**

Creates a Commodity Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
BuySell	Takes the values of either 'Buy' or 'Sell'
commodity_type	Takes the values of 'Oil/Gas', 'Silver', 'Electricity' etc.

**Value**

An object of type Commodity

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

Basel Committee: The standardised approach for measuring counterparty credit risk exposures  
<http://www.bis.org/publ/bcbs279.htm>

**Examples**

```

tr1 = Commodity(Notional=10000,MtM=-50,
BuySell='Buy',SubClass='Energy',commodity_type='Oil')

```



---

 CommodityForward-class

*Commodity Forward Class*


---

## Description

Creates a Commodity Forward Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

## Arguments

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
commodity_type	Takes the values of 'Oil','Gas','Silver','Electricity' etc.
SubClass	Defines the relevant hedging set. Possible values: 'Energy','Agriculture','Metal','Other','Climatic'

## Value

An object of type Commodity Forward

## Author(s)

Tasos Grivas <tasos@openriskcalculator.com>

## References

Regulation (EU) 2019/876 of the European Parliament and of the Council of 20 May 2019 <http://data.europa.eu/eli/reg/2019/876>

## Examples

```
## the Commodity Forward trade given in the Basel regulation Commodity example
tr1 = CommodityForward(Notional=10000,MtM=-50, Si=0, Ei=0.75,
BuySell='Buy', SubClass='Energy', commodity_type='Oil')
```

---

CommSwap-class	<i>Commodity Swap Class</i>
----------------	-----------------------------

---

**Description**

Creates a Commodity Swap Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Value**

An object of type CommSwap

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

Basel Committee: The standardised approach for measuring counterparty credit risk exposures  
<http://www.bis.org/publ/bcbs279.htm>

---

CrossSampleEntropy	<i>Angular distance metrics</i>
--------------------	---------------------------------

---

**Description**

Calculates the cross sample entropy between two track records of various assets/strategies.

**Usage**

```
CrossSampleEntropy(returns_matrix, m = 2, r = 0.2)
```

**Arguments**

returns_matrix	a matrix containing the track records of the underlying assets/strategies. These will be normalized during the algorithm
m	an integer value defining the embedding dimension , default value is 2
r	a double value defining the tolerance, default value is 0.2

**Value**

The value of cross sample entropy

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

<https://physoc.onlinelibrary.wiley.com/doi/epdf/10.1113/expphysiol.2007.037150>

**Examples**

```
## calling CrossSampleEntropy() without an argument loads the historical edhec data
## for the "Short Selling" and "Convertible Arbitrage" strategies
returns_matrix = PerformanceAnalytics::edhec[,c("Short Selling", "Convertible Arbitrage")]
Cross_Sample_Entropy = CrossSampleEntropy(returns_matrix,m=2,r=0.2)
```

---

 CSA-class

*CSA Class*


---

**Description**

Creates a collateral agreement Object containing all the relevant data and methods regarding the maturity factor and the calculation of the exposures after applying the relevant threshold

**Arguments**

ID	The ID of the CSA ID
Counterparty	The counterparty the CSA is linked to
Currency	The currency that the CSA applies to (can be a list of different currencies)
TradeGroups	The trade groups that the CSA applies to
Values_type	The type of the numerical values (can be "Actual" or "Perc" whereby the values are percentages of the MtM)
thres_cpty	The maximum exposure that the counterparty can generate before collateral will need to be posted
thres_PO	The maximum exposure that the processing organization can generate before collateral will need to be posted
MTA_cpty	The minimum transfer amount for the counterparty
MTA_PO	The minimum transfer amount for the processing organization
IM_cpty	The initial margin that is posted by the counterparty
IM_PO	The initial margin that is posted by the processing organization
mpor_days	The margin period of risk in days
remargin_freq	The frequency of re-margining the exposure in days
rounding	The rounding amount of the transfers

**Value**

An object of type CSA

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

Basel Committee: The standardised approach for measuring counterparty credit risk exposures  
<http://www.bis.org/publ/bcbs279.htm>

**Examples**

```

csa_raw = read.csv(system.file("extdata", "CSA.csv", package = "Trading"),
header=TRUE,stringsAsFactors = FALSE)

csas = list()
for(i in 1:nrow(csa_raw))
{
  csas[[i]] = CSA()
  csas[[i]]$PopulateViaCSV(csa_raw[i,])
}

```

---

Curve-class

*Curve Class*


---

**Description**

Creates a Curve Object containing pairs of Tenors with relevant rates and the interpolation function. Also, methods for populating the object via a .csv file and the generation of the interpolation function via cubic splines are included.

**Arguments**

Tenors	The Tenors of the curve
Rates	The rates on the corresponding tenors
interp_function	(Optional) The interpolation function of the curve. Can be populated via the 'CalcInterpPoints' method

**Value**

An object of type Curve

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
## generating a curve either directly or through a csv -
## the spot_rates.csv file can be found on the extdata folder in the installation library path
funding_curve = Curve(Tenors=c(1,2,3,4,5,6,10),Rates=c(4,17,43,47,76,90,110))
spot_rates = Curve()
spot_rates$PopulateViaCSV('spot_rates.csv')
time_points = seq(0,5,0.01)
spot_curve    = spot_rates$CalcInterpPoints(time_points)
```

---

DynamicBeta

*Time Varying Beta via Kalman filter & smoother*


---

**Description**

Calculates the beta of an investment strategy or stock by applying the Kalman filter & smoother. Apart from the beta timeseries, the state covariances are also returned so as to provide an estimate of the uncertainty of the results. The python package "Pykalman" is used for the calculations given its proven stability.

**Usage**

```
DynamicBeta(csvfilename, do_not_set_to_true = FALSE)
```

**Arguments**

```
csvfilename    the name of csv file containing the track record of the fund & the benchmark
do_not_set_to_true
                function returns zero when TRUE - used only so as to pass the CRAN tests
                where pykalman couldn't be installed
```

**Value**

A list of beta values based on Kalman Filter & smoother and the respective covariance matrices

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
## calling DynamicBeta() without an argument loads a test file containing a sample track
## record and a benchmark index
## ATTENTION!!!: set do_not_set_to_true to FALSE when running the example
##-- this is only used to pass CRAN tests whereby
## pykalman was not installable!
dyn_beta_values = DynamicBeta(do_not_set_to_true = TRUE)
```

---

 Equity-class

*Equity Class*


---

### Description

Creates an Equity object

### Arguments

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
BuySell	Takes the values of either 'Buy' or 'Sell'
ISIN	the ISIN of the Equity
traded_price	the price that trade was done
Issuer	the issuer of the stock

### Value

An object of type Equity

### Author(s)

Tasos Grivas <tasos@openriskcalculator.com>

### Examples

```
tr1 = Equity(external_id="ext1",Notional=10000,MtM=30,Currency="EUR",BuySell='Buy',
traded_price = 10,ISIN = "XS04340432",Issuer='Firma')
```

---

 EquityIndexFuture-class

*Equity Index Future Class*


---

### Description

Creates an Equity Index Future object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
traded_price	the price that trade was done

**Value**

An object of type EquityIndexFuture

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
example_trades = ParseTrades()
Equity_Index_Future_trade = example_trades[[18]]
```

---

EquityOptionIndex-class

*Equity Option Index Class*

---

**Description**

Creates an Equity Option Index object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
traded_price	the price that trade was done

**Value**

An object of type EquityOption

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

---

EquityOptionSingle-class

*Equity Option Single Class*

---

**Description**

Creates an Equity Option Single object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
traded_price	the price that trade was done

**Value**

An object of type EquityOption

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

---

EuroJackpotExample

*Eurojackpot analysis example*

---

**Description**

Displays how the functionality related to the eurojackpot analysis can be utilized

**Usage**

EuroJackpotExample()

**Value**

The final results



**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
# This software is covered by GPL license and provided strictly for educational
# reasons (no actual investment/betting decisions should be taken based on this)

final_results = EuroJackpotExample()
```

---

EuroJackpotResults      *Returns all the EuroJackpot results until the end of 2023*

---

**Description**

Returns all the EuroJackpot results since the first draw on Feb 2004 until the end of 2023

**Usage**

```
EuroJackpotResults()
```

**Value**

A dataframe with all the EuroJackpot results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
eurojackpot_results = EuroJackpotResults()
```

---

EuroLotteryAllCombinations  
*Returns all the possible number combinations for EuroMillions/EuroJackpot*

---

**Description**

Returns all the possible number combinations for EuroMillions/EuroJackpot

**Usage**

```
EuroLotteryAllCombinations()
```

**Value**

PnL figures

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
# returns all the 139,838,160 possible combinations, can create memory issues.  
# all_combinations = EuroLotteryAllCombinations()
```

---

EuroLotteryBacktesting

*Euromillions/EuroJackpot Backtesting*

---

**Description**

Backtests the numbers the user has selected against the full (or the specified) history of Euromillions/EuroJackpot results

**Usage**

```
EuroLotteryBacktesting(euroLottery_results, date_since, user_input)
```

**Arguments**

euroLottery_results	The full list of EuroMillions/EuroJackpot results
date_since	The date after which the analysis is to be performed, i.e. 2022-12-22
user_input	The seven numbers the user has selected

**Value**

The backtested results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
euromillions_results = EuroMillionsResults()  
user_input = c(10,20,30,40,50,5,10)  
backtested_results = EuroLotteryBacktesting(euromillions_results, '2005-01-01', user_input)
```

---

EuroMillionsExample    *Euromillions analysis example*

---

**Description**

Displays how the functionality related to the euromillions analysis can be utilized

**Usage**

```
EuroMillionsExample()
```

**Value**

The final results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
# This software is covered by GPL license and provided strictly for educational  
# reasons (no actual investment/betting decisions should be taken based on this)  
  
final_results = EuroMillionsExample()
```

---

EuroMillionsResults    *Returns all the EuroMillions results until the end of 2023*

---

**Description**

Returns all the EuroMillions results since the first draw on Feb 2004 until the end of 2023

**Usage**

```
EuroMillionsResults()
```

**Value**

A dataframe with all the EuroMillions results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
euromillions_results = EuroMillionsResults()
```

---

FxForward-class	<i>Fx Forward Class</i>
-----------------	-------------------------

---

**Description**

Creates a FX Forward Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency that the input amounts are in
ccyPair	The currency Pair of the trade
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
traded_price	the price that trade was done

**Value**

An object of type FX Forward

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

Basel Committee: The standardised approach for measuring counterparty credit risk exposures  
<http://www.bis.org/publ/bcbs279.htm>

**Examples**

```
## an FX Forward trade
tr1 = FxForward(Notional=10000,MtM=-50, Si=0, Ei=0.75, BuySell='Buy', ccyPair="EUR/USD")
## a dynamic version of the same trade
tr2 = FxForward(MtM=-50, Si=0, Ei=0.75, ccy_paying="USD", amount_paying=10000,
ccy_receiving="EUR", amount_receiving=9900)
tr2$base_ccy="EUR"
tr2$setFXDynamic()
```

FxSwap-class

*Fx Swap Class***Description**

Creates an FX Swap object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency that the input amounts are in
ccyPair	The currency Pair of the trade
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
traded_price	the price that trade was done
fx_near_leg_fields	(Optional) In case the near leg hasn't settled yet, its notional, MtM, settlement date should be provided separated via a semicolon

**Value**

An object of type FXSwap

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

Basel Committee: The standardised approach for measuring counterparty credit risk exposures  
<http://www.bis.org/publ/bcbs279.htm>

**Examples**

```
tr1 = FxSwap(Notional=10000,MtM=30,ccyPair="EUR/USD",Si=0,Ei=10,
BuySell='Buy',fx_near_leg_fields='1000;-20;2020-02-11')
```

---

GetTradeDetails	<i>Returns a list with the populated fields of a Trade Object</i>
-----------------	---

---

**Description**

Returns a list with the populated fields of a Trade Object

**Usage**

```
GetTradeDetails(trade)
```

**Arguments**

trade	A trade Object
-------	----------------

**Value**

A list of fields

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
example_trades = ParseTrades()
Equity_Index_Future_trade = example_trades[[18]]
populated_fields = GetTradeDetails(Equity_Index_Future_trade)
```

---

HashTable-class	<i>Hashtable Class</i>
-----------------	------------------------

---

**Description**

Creates a hashtable-like object so as to represent data with a key structure (for example addon tables, rating-based factors etc). Also, it includes methods for populating the object via a .csv file and finding a value based on a specific key on an interval of keys For examples of the format of the CSVs files, please view RatingsMapping.csv or AddonTable.csv on the extdata folder in the installation folder of the library

**Arguments**

keys	A vector of keys
values	A vector of values mapping to the keys
keys_type	The type of the keys
values_type	The type of the values

**Value**

An object of type HashTable

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
## loading a ratings' mapping matrix from the extdata folder
rating_table = HashTable('RatingsMapping.csv', "character", "numeric")
reg_weight =rating_table$FindValue("AAA")
```

---

InformationAdjustedBeta

*Information Adjusted Beta*

---

**Description**

Calculates the Information-Adjusted Beta between the track records of two assets/strategies which covers for cases whereby the 'typical' linearity and Gaussian I.I.D assumptions do not hold. The normalized cross sample entropy has been utilized for the mutual information estimation.

**Usage**

```
InformationAdjustedBeta(x, y, m = 2, r = 0.2)
```

**Arguments**

x	a vector containing the track record of the underlying asset/strategy (can be a data.table, data.frame, vector etc)
y	a vector containing the track record of the underlying asset/strategy (can be a data.table, data.frame, vector etc)
m	an integer value defining the embedding dimension for the sample entropy calculation, default value is 2
r	a double value defining the tolerance for the sample entropy calculation, default value is 0.2

**Value**

The information adjusted Beta

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

## References

[https://github.com/devisechain/Devise/blob/master/yellow\\_paper.pdf](https://github.com/devisechain/Devise/blob/master/yellow_paper.pdf)

## Examples

```
x = PerformanceAnalytics::edhec[,c("Short Selling")]
y = PerformanceAnalytics::edhec[,c("Convertible Arbitrage")]
Information_Adjusted_Beta = InformationAdjustedBeta(x, y, m=2, r=0.2)
```

---

InformationAdjustedCorr

*Information Adjusted Correlation*

---

## Description

Calculates the Information-Adjusted Correlation between the track records of various assets/strategies which covers for cases whereby the 'typical' Pearson's correlation assumptions do not hold. The normalized cross sample entropy has been utilized for the mutual information estimation.

## Usage

```
InformationAdjustedCorr(x, y, m = 2, r = 0.2)
```

## Arguments

x	a vector containing the track record of the underlying asset/strategy (can be a data.table, data.frame, vector etc)
y	a vector containing the track record of the underlying asset/strategy (can be a data.table, data.frame, vector etc)
m	an integer value defining the embedding dimension for the sample entropy calculation, default value is 2
r	a double value defining the tolerance for the sample entropy calculation, default value is 0.2

## Value

The information adjusted correlation

## Author(s)

Tasos Grivas <tasos@openriskcalculator.com>

## References

[https://github.com/devisechain/Devise/blob/master/yellow\\_paper.pdf](https://github.com/devisechain/Devise/blob/master/yellow_paper.pdf)



**Examples**

```
x = PerformanceAnalytics::edhec[,c("Short Selling")]
y = PerformanceAnalytics::edhec[,c("Convertible Arbitrage")]
Information_Adjusted_Corr = InformationAdjustedCorr(x, y, m=2, r=0.2)
```

---

IRDFuture-class	<i>IRD Future Class</i>
-----------------	-------------------------

---

**Description**

Creates an IRD Future Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'

**Value**

An object of type IRDFuture

---

IRDSwap-class	<i>IRD Swap Class</i>
---------------	-----------------------

---

**Description**

Creates an IRD Swap Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'

**Value**

An object of type IRDSwap

**Examples**

```
# the IRD Swap trade given in the Basel regulation IRD example
tr1 = IRDSwap(Notional=10000,MtM=30,Currency="USD",Si=0,Ei=10,BuySell='Buy')
```

---

IRDSwaption-class      *IRD Swaption Class*

---

**Description**

Creates an IRD Swaption Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
OptionType	Takes the values of either 'Put' or 'Call'
UnderlyingPrice	The current price of the underlying
StrikePrice	The strike price of the option

**Value**

An object of type IRDSwaption

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

Basel Committee: The standardised approach for measuring counterparty credit risk exposures  
<http://www.bis.org/publ/bcbs279.htm>

**Examples**

```
# the Swaption trade given in the Basel regulation IRD example
tr3 = IRDSwaption(Notional=5000,MtM=50,Currency="EUR",Si=1,Ei=11,BuySell='Sell',
OptionType='Put',UnderlyingPrice=0.06,StrikePrice=0.05)
```

---

IRDSwapVol-class	<i>IRD Swap Volatility Class</i>
------------------	----------------------------------

---

**Description**

Creates an IRD Swap Volatility-based Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Value**

An object of type IRDSwapVol

---

martingale_strategy_repetitions	<i>Martingale Strategy Repetitions</i>
---------------------------------	--

---

**Description**

Calculates the number of repetitions needed for a specific number of consecutive failed trades/bet to appear. This can apply to roulette betting but also trading algorithms which use the same logic on doubling down after a failed trade.

**Usage**

```
martingale_strategy_repetitions(
  length_of_targeted_sequence,
  prob_of_success = 18/37,
  simulations_num,
  trials_per_sim,
  quantile_perc
)
```

**Arguments**

length_of_targeted_sequence	The number of consecutive failed trades/bets that we try to calculate the expected number of repetitions for
prob_of_success	The probability of a successful trade/bet
simulations_num	The number of simulations to be run
trials_per_sim	The number of trials in each simulation
quantile_perc	(Optional) When set, the number of repetitions expected with such probability is returned.

**Value**

A list containing the number of repetitions needed to reach the targeted sequence for the first time in each simulation (will be zero if the sequence is not found) and, when the `quantile_perc` is set, the above number of repetitions.

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

[https://en.wikipedia.org/wiki/Roulette#Betting\\_strategies\\_and\\_tactics](https://en.wikipedia.org/wiki/Roulette#Betting_strategies_and_tactics)

**Examples**

```
# This software is covered by GPL license and provided strictly for educational
# reasons (no actual investment or betting decisions should be taken based on this)
# On top of these, the below example contains a tiny number of simulations and
# trials just to pass CRAN tests - the user would have to highly increase both
# variables when running these.
repetitions_for_failed_sequence = martingale_strategy_repetitions(length_of_targeted_sequence = 8,
prob_of_success = 18/37, simulations_num = 1000, trials_per_sim = 10000, quantile_perc = 0.1)
repetitions_for_failed_sequence$relevant_quantile
summary(repetitions_for_failed_sequence$num_of_trials_needed)
```

---

NormXASampEn

*Normalized Cross Sample Entropy*

---

**Description**

Calculates the Normalized Cross Sample Entropy of the track records of two assets/strategies based on the sample entropy.

**Usage**

```
NormXASampEn(x, y, m = 2, r = 0.2)
```

**Arguments**

x	a vector containing the track record of the underlying asset/strategy, this will be normalized during the algorithm
y	a vector containing the track record of the underlying asset/strategy, this will be normalized during the algorithm
m	an integer value defining the embedding dimension , default value is 2
r	a double value defining the tolerance, default value is 0.2

**Value**

A value containing the NormXASampEn

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

Lopez de Prado, Marcos, Codependence (Presentation Slides) (January 2, 2020). Available at SSRN: <https://ssrn.com/abstract=3512994>

**Examples**

```
x = PerformanceAnalytics::edhec[,c("Short Selling")]
y = PerformanceAnalytics::edhec[,c("Convertible Arbitrage")]
Normalized_Cross_Sample_Entropy = NormXASampEn(x, y, m=2, r=0.2)
```

---

OtherExposure-class    *OtherExposure Class*

---

**Description**

Creates a OtherExposure Object with the relevant info needed to calculate the Exposure-at-Default (EAD)

**Arguments**

Notional	The notional amount of the trade
MTM	The mark-to-market valuation of the trade
Currency	The currency set that the trade belongs to
Si	The number of years that the trade will take to start (zero if already started)
Ei	The number of years that the trade will expire
BuySell	Takes the values of either 'Buy' or 'Sell'
SubClass	Defines the hedging set the relevant trade will belong to

**Value**

An object of type OtherExposure

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

## References

Regulation (EU) 2019/876 of the European Parliament and of the Council of 20 May 2019 <http://data.europa.eu/eli/reg/2019/876>

## Examples

```
tr1 = OtherExposure(Notional=10000,MtM=-50,Si=0,Ei=10,
BuySell='Buy',SubClass='Other_1')
```

---

OuterJoinMerge	<i>Returns all possible combinations of two dataframes</i>
----------------	--

---

## Description

Returns all possible combinations of two dataframes

## Usage

```
OuterJoinMerge(df_a, df_b)
```

## Arguments

df_a	The first dataframe
df_b	The second dataframe

## Value

A dataframe with all combinations

## Author(s)

Tasos Grivas <[tasos@openriskcalculator.com](mailto:tasos@openriskcalculator.com)>

## Examples

```
df_a = data.frame(matrix(seq(1,20),nrow = 5, ncol = 4))
df_b = data.frame(matrix(seq(21,40),nrow = 5, ncol = 4))
joined_df = OuterJoinMerge(df_a, df_b)
```

---

ParseTrades	<i>Parse trades through a .csv file.</i>
-------------	--

---

**Description**

Parse trades through a .csv file. In case no file name is given, an example file is automatically loaded containing trades corresponding to Basel's SA-CCR regulation (the example trades file can be found on the extdata folder in the installation library path)

**Usage**

```
ParseTrades(csvfilename)
```

**Arguments**

csvfilename      the name of csv file containing the trades

**Value**

A list of trades

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
## calling ParseTrades() without an argument loads a test file containing all
## the different trade types supported
example_trades = ParseTrades()
```

---

roulette_pl_calculator_dalembert	<i>Roulette P&amp;L betting based on the D'Alembert Betting System</i>
----------------------------------	--

---

**Description**

Calculates the potential profit or loss when someone is betting in the roulette based on the D'Alembert Betting System

**Usage**

```
roulette_pl_calculator_dalembert(  
  bet_minimum,  
  bet_maximum,  
  initial_capital,  
  simulations_num,  
  trials_per_sim  
)
```

**Arguments**

bet_minimum	The minimum betting amount that the casino allows
bet_maximum	The maximum betting amount that the casino allows
initial_capital	The initial capital to be used
simulations_num	The number of simulations to be run
trials_per_sim	The number of trials in each simulation

**Value**

A list containing the minimum, the maximum and the final balance for each simulation. Also the P&L graph for the last simulation will be plotted.

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

[https://en.wikipedia.org/wiki/Roulette#Betting\\_strategies\\_and\\_tactics](https://en.wikipedia.org/wiki/Roulette#Betting_strategies_and_tactics)

**Examples**

```
# This software is covered by GPL license and provided strictly for educational  
# reasons (no actual investment/betting decisions should be taken based on this)  
# On top of these, the below example contains a tiny number of simulations and  
# trials just to pass CRAN tests - the user would have to highly increase both  
# variables when running these.  
pl_results = roulette_pl_calculator_dalembert(bet_minimum = 0.1 , bet_maximum = 3276.8,  
initial_capital = 20000, simulations_num = 100, trials_per_sim = 100)  
summary(pl_results$min_capital)  
summary(pl_results$max_capital)  
summary(pl_results$final_capital)
```



---

roulette\_pl\_calculator\_fibonacci

*Roulette P&L betting based on the Fibonacci Betting System*

---

## Description

Calculates the potential profit or loss when someone is betting in the roulette based on the Fibonacci Betting System.

## Usage

```
roulette_pl_calculator_fibonacci(
    bet_minimum,
    bet_maximum,
    initial_capital,
    simulations_num,
    trials_per_sim
)
```

## Arguments

bet_minimum	The minimum betting amount that the casino allows
bet_maximum	The maximum betting amount that the casino allows
initial_capital	The initial capital to be used
simulations_num	The number of simulations to be run
trials_per_sim	The number of trials in each simulation

## Value

A list containing the minimum, the maximum and the final balance for each simulation. Also the P&L graph for the last simulation will be plotted.

## Author(s)

Tasos Grivas <tasos@openriskcalculator.com>

## References

[https://en.wikipedia.org/wiki/Roulette#Betting\\_strategies\\_and\\_tactics](https://en.wikipedia.org/wiki/Roulette#Betting_strategies_and_tactics)

**Examples**

```
# This software is covered by GPL license and provided strictly for educational
# reasons (no actual investment or betting decisions should be taken based on this)
# On top of these, the below example contains a tiny number of simulations and
# trials just to pass CRAN tests - the user would have to highly increase both
# variables when running these.
pl_results = roulette_pl_calculator_fibonacci(bet_minimum = 0.1 , bet_maximum = 6000,
  initial_capital = 20000, simulations_num = 100, trials_per_sim = 100)
summary(pl_results$min_capital)
summary(pl_results$max_capital)
summary(pl_results$final_capital)
```

---

```
roulette_pl_calculator_labouchere
```

*Roulette P&L betting based on the Labouchere Betting System*

---

**Description**

Calculates the potential profit or loss when someone is betting in the roulette based on the Labouchere Betting System.

**Usage**

```
roulette_pl_calculator_labouchere(
  bet_minimum,
  bet_maximum,
  initial_capital,
  profit_target,
  profit_sequence,
  simulations_num,
  trials_per_sim
)
```

**Arguments**

bet_minimum	The minimum betting amount that the casino allows
bet_maximum	The maximum betting amount that the casino allows
initial_capital	The initial capital to be used
profit_target	The profit amount to be earned
profit_sequence	(Optional) the amounts of the bets to reach this profit amount. If omitted, the minimum betting amount will be used
simulations_num	The number of simulations to be run
trials_per_sim	The number of trials in each simulation

**Value**

A list containing the minimum, the maximum and the final balance for each simulation. Also the P&L graph for the last simulation will be plotted.

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

[https://en.wikipedia.org/wiki/Roulette#Betting\\_strategies\\_and\\_tactics](https://en.wikipedia.org/wiki/Roulette#Betting_strategies_and_tactics)

**Examples**

```
# This software is covered by GPL license and provided strictly for educational
# reasons (no actual investment/betting decisions should be taken based on this)
# On top of these, the below example contains a tiny number of simulations and
# trials just to pass CRAN tests - the user would have to highly increase both
# variables when running these.
pl_results = roulette_pl_calculator_labouchere(bet_minimum = 0.1 , bet_maximum = 3276.8,
initial_capital = 20000, profit_target = 100, profit_sequence = rep(10,10),
simulations_num = 100, trials_per_sim = 100)
summary(pl_results$min_capital)
summary(pl_results$max_capital)
summary(pl_results$final_capital)
```

---

roulette\_pl\_calculator\_martingale

*Roulette P&L betting based on a modified martingale strategy*

---

**Description**

Calculates the potential profit or loss when someone is betting in the roulette based on the martingale system while trying to reduce the risk by 1. Starting to double after the first loss 2. Not doubling if the second number is zero.

**Usage**

```
roulette_pl_calculator_martingale(
  bet_minimum,
  bet_maximum,
  initial_capital,
  simulations_num,
  trials_per_sim
)
```

**Arguments**

bet\_minimum      The minimum betting amount that the casino allows  
bet\_maximum      The maximum betting amount that the casino allows  
initial\_capital      The initial capital to be used  
simulations\_num      The number of simulations to be run  
trials\_per\_sim      The number of trials in each simulation

**Value**

A list containing the minimum, the maximum and the final balance for each simulation. Also the P&L graph for the last simulation will be plotted.

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

[https://en.wikipedia.org/wiki/Roulette#Betting\\_strategies\\_and\\_tactics](https://en.wikipedia.org/wiki/Roulette#Betting_strategies_and_tactics)

**Examples**

```
# This software is covered by GPL license and provided strictly for educational
# reasons (no actual investment/betting decisions should be taken based on this)
# On top of these, the below example contains a tiny number of simulations and
# trials just to pass CRAN tests - the user would have to highly increase both
# variables when running these.
pl_results = roulette_pl_calculator_martingale(bet_minimum = 0.1 , bet_maximum = 3276.8,
initial_capital = 20000, simulations_num = 100, trials_per_sim = 100)
summary(pl_results$min_capital)
summary(pl_results$max_capital)
summary(pl_results$final_capital)
```

---

roulette\_pl\_calculator\_specific\_number

*Roulette P&L betting on a specific number*

---

**Description**

Calculates the potential profit or loss when someone is betting on a specific number in the roulette and keeps doubling every eighteen spins if the number hasn't appeared yet.

### Usage

```
roulette_pl_calculator_specific_number(
  bet_minimum,
  bet_maximum,
  initial_capital,
  targeted_number,
  simulations_num,
  trials_per_sim,
  stop_loss
)
```

### Arguments

bet_minimum	The minimum betting amount that the casino allows
bet_maximum	The maximum betting amount that the casino allows
initial_capital	The initial capital to be used
targeted_number	The specific number that we expect to be drawn (statistically speaking, this should have zero effect on the results)
simulations_num	The number of simulations to be run
trials_per_sim	The number of trials in each simulation
stop_loss	(Optional) The number of spins after which the betting amount will go back to the minimum if the targeted number hasn't appeared.

### Value

A list containing the minimum, the maximum and the final balance for each simulation. Also the P&L graph for the last simulation will be plotted.

### Author(s)

Tasos Grivas <tasos@openriskcalculator.com>

### References

[https://en.wikipedia.org/wiki/Roulette#Betting\\_strategies\\_and\\_tactics](https://en.wikipedia.org/wiki/Roulette#Betting_strategies_and_tactics)

### Examples

```
# This software is covered by GPL license and provided strictly for educational
# reasons (no actual investment or betting decisions should be taken based on this)
# On top of these, the below example contains a tiny number of simulations and
# trials just to pass CRAN tests - the user would have to highly increase both
# variables when running these.
pl_results = roulette_pl_calculator_specific_number(bet_minimum = 0.1 , bet_maximum = 3276.8,
initial_capital = 20000, targeted_number = 0, simulations_num = 100,
```

```

trials_per_sim = 100, stop_loss = 180)
summary(pl_results$min_capital)
summary(pl_results$max_capital)
summary(pl_results$final_capital)

```

---

SampleEntropy

*Sample Entropy*


---

### Description

Calculates the sample entropy of a track record. Sample entropy is an improvement of the approximate entropy and should produce accurate results for timeseries of smaller length like historical returns of strategies

### Usage

```
SampleEntropy(returns, m = 2, r = 0.2)
```

### Arguments

returns	a vector containing the track record of the underlying asset/strategy, these will be normalized during the algorithm
m	an integer value defining the embedding dimension , default value is 2
r	a double value defining the tolerance, default value is 0.2

### Value

The sample Entropy of the input returns

### Author(s)

Tasos Grivas <tasos@openriskcalculator.com>

### References

[https://en.wikipedia.org/wiki/Sample\\_entropy](https://en.wikipedia.org/wiki/Sample_entropy)

### Examples

```

## calling SampleEntropy() without an argument loads the historical edhec
## data for the "Short Selling" strategy
returns = PerformanceAnalytics::edhec[,c("Short Selling")]
Sample_Entropy = SampleEntropy(returns,m=2,r=0.2)

```

---

SelectDerivatives      *Select the derivatives out of a trades' list*

---

**Description**

Select the derivatives out of a trades' list which will be utilized to calculate the CCR Exposure.

**Usage**

```
SelectDerivatives(trades_list)
```

**Arguments**

trades\_list      the file holding the trades of the portfolio

**Value**

The derivatives out of a trades' list

**Author(s)**

Tasos Grivas <info@openriskcalculator.com>

**References**

Regulation (EU) 2019/876 of the European Parliament and of the Council of 20 May 2019 <http://data.europa.eu/eli/reg/2019/876>

---

SetForLifeBacktesting      *Set For Life Backtesting*

---

**Description**

Backtests the numbers the user has selected against the full (or the specified) history of Set For Life results

**Usage**

```
SetForLifeBacktesting(setforlife_results, date_since, user_input)
```

**Arguments**

setforlife\_results      The full list of Set For Life results

date\_since      The date after which the analysis is to be performed, i.e. 2022-12-22

user\_input      The seven numbers the user has selected

**Value**

The backtested results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
setforlife_results = SetForLifeResults()  
user_input = c(10,20,30,40,50,5,10)  
backtested_results = SetForLifeBacktesting(setforlife_results, '2005-01-01', user_input)
```

---

SetForLifeExample      *Set For Life analysis example*

---

**Description**

Displays how the functionality related to the Set For Life analysis can be utilized

**Usage**

```
SetForLifeExample()
```

**Value**

The final results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
# This software is covered by GPL license and provided strictly for educational  
# reasons (no actual investment/betting decisions should be taken based on this)  
  
final_results = SetForLifeExample()
```



---

SetForLifeResults	<i>Returns all the EuroJackpot results until the end of 2023</i>
-------------------	--

---

**Description**

Returns all the SetForLifeResults results since the first draw on Feb 2004 until the end of 2023

**Usage**

```
SetForLifeResults()
```

**Value**

A dataframe with all the EuroJackpot results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
eurojackpot_results = EuroJackpotResults()
```

---

top5	<i>Top 5 most or least lucky numbers for EuroMillions/EuroJackpot</i>
------	---

---

**Description**

Returns the top 5 most or least lucky euromillion numbers

**Usage**

```
top5(eurolottery_results, date_since, least_lucky = FALSE)
```

**Arguments**

eurolottery_results	The full list of EuroMillions/EuroJackpot results
date_since	The date after which the analysis is to be performed, i.e. 2022-12-22
least_lucky	If TRUE, the least lucky numbers will be returned (default FALSE)

**Value**

Top 5 numbers

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
euromillions_results = EuroMillionsResults()  
top_5 = top5(euromillions_results, '2022-12-22', least_lucky=TRUE)
```

---

Total\_Carbon\_Emissions

*Total Carbon Emissions*

---

**Description**

Returns the absolute greenhouse gas emissions associated with a portfolio, expressed in tons CO<sub>2</sub>e. Under this approach, if an investor owns 5 percent of a company's total market capitalization, then the investor owns 5 percent of the company as well as 5 percent of the company's GHG (or carbon) emissions.

**Usage**

```
Total_Carbon_Emissions(portfolio_exposure, emissions_capitalization_data)
```

**Arguments**

portfolio\_exposure

The exposure per issuer in the portfolio

emissions\_capitalization\_data

The capitalization and the Scope 1 & 2 GHG emissions per issuer

**Value**

The absolute greenhouse gas emissions associated with a portfolio, expressed in tons CO<sub>2</sub>e

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**References**

<https://www.tcfhub.org/Downloads/pdfs/E09>

**Examples**

```
portfolio_exposure = data.table::data.table(Issuers = c('A','B','C'),
exposures = c(100, 200, 50))
emissions_capitalization_data = data.table::data.table(Issuers = c('A','B','C'),
emissions = c(1000, 5000, 6000),
Capitalization = c(20000, 10000, 30000))
Total_Carbon_Emissions(portfolio_exposure, emissions_capitalization_data)
```

---

UKLotteryBacktesting *UKLottery Backtesting*

---

**Description**

Backtests the numbers the user has selected against the full (or the specified) history of UKLottery results

**Usage**

```
UKLotteryBacktesting(uklottery_results, date_since, user_input)
```

**Arguments**

uklottery_results	The full list of UKLottery results
date_since	The date after which the analysis is to be performed, i.e. 2022-12-22
user_input	The seven numbers the user has selected

**Value**

The backtested results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
uklottery_results = UKLotteryResults()
user_input = c(5,10,20,30,40,50)
backtested_results = UKLotteryBacktesting(uklottery_results, '2005-01-01', user_input)
```

UKLotteryExample      *UK Lottery analysis example*

---

**Description**

Displays how the functionality related to the UK Lottery analysis can be utilized

**Usage**

```
UKLotteryExample()
```

**Value**

The final results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
# This software is covered by GPL license and provided strictly for educational
# reasons (no actual investment/betting decisions should be taken based on this)

final_results = UKLotteryExample()
```

---

UKLotteryResults      *Returns all the UKLottery results until the beginning of 2025*

---

**Description**

Returns all the UKLottery results since the first draw on Nov 1994 until the beginning of 2025

**Usage**

```
UKLotteryResults()
```

**Value**

A dataframe with all the UKLottery results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
UKLotto_results = UKLotteryResults()
```

---

UKThunderballBacktesting

*UK ThunderBall Backtesting*

---

**Description**

Backtests the numbers the user has selected against the full (or the specified) history of UK ThunderBall results

**Usage**

```
UKThunderballBacktesting(ukthunderball_results, date_since, user_input)
```

**Arguments**

ukthunderball_results	The full list of UK ThunderBall results
date_since	The date after which the analysis is to be performed, i.e. 2022-12-22
user_input	The seven numbers the user has selected

**Value**

The backtested results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
ukthunderball_results = UKThunderBallResults()  
user_input = c(10,20,30,31,32,5)  
backtested_results = UKThunderballBacktesting(ukthunderball_results, user_input = user_input)
```

---

UKThunderballExample *UK ThunderBall analysis example*

---

**Description**

Displays how the functionality related to the UK ThunderBall analysis can be utilized

**Usage**

```
UKThunderballExample()
```

**Value**

The final results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
# This software is covered by GPL license and provided strictly for educational
# reasons (no actual investment/betting decisions should be taken based on this)

final_results = EuroJackpotExample()
```

---

UKThunderBallResults *Returns all the UK ThunderBall results until the beginning of 2025*

---

**Description**

Returns all the UK ThunderBall results until the beginning of 2025

**Usage**

```
UKThunderBallResults()
```

**Value**

A dataframe with all the UK ThunderBall results

**Author(s)**

Tasos Grivas <tasos@openriskcalculator.com>

**Examples**

```
UKThunderBall_Results = UKThunderBallResults()
```

---

VariationOfInformation

*Variation of Information*

---

### Description

Calculates the variation of information of the track records of two assets/strategies based on the sample entropy.

### Usage

```
VariationOfInformation(x, y, m = 2, r = 0.2, normalized = TRUE)
```

### Arguments

x	a vector containing the track record of the underlying asset/strategy, this will be normalized during the algorithm
y	a vector containing the track record of the underlying asset/strategy, this will be normalized during the algorithm
m	an integer value defining the embedding dimension , default value is 2
r	a double value defining the tolerance, default value is 0.2
normalized	a boolean value so as to bound the return value between 0 and 1, default value is TRUE

### Value

A value containing the variation of information

### Author(s)

Tasos Grivas <tasos@openriskcalculator.com>

### References

Lopez de Prado, Marcos, Codependence (Presentation Slides) (January 2, 2020). Available at SSRN: <https://ssrn.com/abstract=3512994>

### Examples

```
x = PerformanceAnalytics::edhec[,c("Short Selling")]
y = PerformanceAnalytics::edhec[,c("Convertible Arbitrage")]
variation_of_information = VariationOfInformation(x, y, m=2, r=0.2, normalized = TRUE)
```

---

Weighted\_Average\_Carbon\_Intensity

*Weighted Average Carbon Intensity*

---

### Description

Returns the portfolio's exposure to each issuer expressed in tons CO<sub>2</sub>e / \$M revenue. Scope 1 and Scope 2 GHG emissions are allocated based on portfolio weights (the current value of the investment relative to the current portfolio value), rather than the equity ownership approach

### Usage

```
Weighted_Average_Carbon_Intensity(portfolio_exposure, emissions_revenue_data)
```

### Arguments

portfolio\_exposure

The exposure per issuer in the portfolio

emissions\_revenue\_data

The capitalization, revenue and the Scope 1 & 2 GHG emissions per issuer

### Value

Total carbon emissions for a portfolio normalized by the market value of the portfolio, expressed in tons CO<sub>2</sub>e / \$M invested.

### Author(s)

Tasos Grivas <tasos@openriskcalculator.com>

### References

<https://www.tcfhub.org/Downloads/pdfs/E09>

### Examples

```
portfolio_exposure = data.table::data.table(Issuers = c('A','B','C'),
  exposures = c(100, 200, 50))
emissions_revenue_data = data.table::data.table(Issuers = c('A','B','C'),
  emissions = c(1000, 5000, 2000),
  revenue = c(2000, 5000, 3000))
Weighted_Average_Carbon_Intensity(portfolio_exposure, emissions_revenue_data)
```



# Index

AngularDistance, 4

Bond (Bond-class), 5  
Bond-class, 5  
BondFuture (BondFuture-class), 6  
BondFuture-class, 6

CalcEuroLotteryPnL, 7  
CalcSetForLifePnL, 7  
CalcUKLotteryPnL, 8  
CalcUKThunderBallPnL, 9  
capped\_fibonacci\_seq, 10  
Carbon\_Footprint, 10  
Carbon\_Intensity, 11  
CDOTranche (CDOTranche-class), 12  
CDOTranche-class, 12  
CDS (CDS-class), 13  
CDS-class, 13  
CDX (CDX-class), 14  
CDX-class, 14  
Chebyshev\_distance, 14  
Collateral (Collateral-class), 15  
Collateral-class, 15  
Commodity (Commodity-class), 16  
Commodity-class, 16  
CommodityForward  
(CommodityForward-class), 17  
CommodityForward-class, 17  
CommSwap (CommSwap-class), 18  
CommSwap-class, 18  
CrossSampleEntropy, 18  
CSA (CSA-class), 19  
CSA-class, 19  
Curve (Curve-class), 20  
Curve-class, 20

DynamicBeta, 21

Equity (Equity-class), 22  
Equity-class, 22

EquityIndexFuture  
(EquityIndexFuture-class), 22  
EquityIndexFuture-class, 22  
EquityOptionIndex  
(EquityOptionIndex-class), 23  
EquityOptionIndex-class, 23  
EquityOptionSingle  
(EquityOptionSingle-class), 24  
EquityOptionSingle-class, 24  
EuroJackpotExample, 24  
EuroJackpotResults, 25  
EuroLotteryAllCombinations, 25  
EuroLotteryBacktesting, 26  
EuroMillionsExample, 27  
EuroMillionsResults, 27

FxForward (FxForward-class), 28  
FxForward-class, 28  
FxSwap (FxSwap-class), 29  
FxSwap-class, 29

GetTradeDetails, 30

HashTable (HashTable-class), 30  
HashTable-class, 30

InformationAdjustedBeta, 31  
InformationAdjustedCorr, 32  
IRDFuture (IRDFuture-class), 33  
IRDFuture-class, 33  
IRDSwap (IRDSwap-class), 33  
IRDSwap-class, 33  
IRDSwaption (IRDSwaption-class), 34  
IRDSwaption-class, 34  
IRDSwapVol (IRDSwapVol-class), 35  
IRDSwapVol-class, 35

martingale\_strategy\_repetitions, 35

NormXASampEn, 36

OtherExposure (OtherExposure-class), [37](#)  
OtherExposure-class, [37](#)  
OuterJoinMerge, [38](#)

ParseTrades, [39](#)

roulette\_pl\_calculator\_dalembert, [39](#)  
roulette\_pl\_calculator\_fibonacci, [41](#)  
roulette\_pl\_calculator\_labouchere, [42](#)  
roulette\_pl\_calculator\_martingale, [43](#)  
roulette\_pl\_calculator\_specific\_number,  
[44](#)

SampleEntropy, [46](#)  
SelectDerivatives, [47](#)  
SetForLifeBacktesting, [47](#)  
SetForLifeExample, [48](#)  
SetForLifeResults, [49](#)

top5, [49](#)  
Total\_Carbon\_Emissions, [50](#)

UKLotteryBacktesting, [51](#)  
UKLotteryExample, [52](#)  
UKLotteryResults, [52](#)  
UKThunderballBacktesting, [53](#)  
UKThunderballExample, [53](#)  
UKThunderBallResults, [54](#)

VariationOfInformation, [55](#)

Weighted\_Average\_Carbon\_Intensity, [56](#)