### autoplot.ffp

*Inspection of a ffp object with ggplot2*

#### Description

Extends the autoplot method for the ffp class.

#### Usage

```r
## S3 method for class 'ffp'
autoplot(object, color = TRUE, ...)

## S3 method for class 'ffp'
plot(object, ...)
```

#### Arguments

- **object**: An object of the ffp class.
- **color**: A logical flag indicating whether (or not) the color argument should be added to the ggplot2 aesthetics.
- **...**: Additional arguments to be passed to autoplot.

#### Value

A ggplot2 object.
**bind_probs**

**Examples**

```r
library(ggplot2)

x <- exp_decay(EuStockMarkets, 0.001)
y <- exp_decay(EuStockMarkets, 0.01)

autoplot(x) +
  scale_color_viridis_c()
autoplot(y) +
  scale_color_viridis_c()
```

---

**bind_probs**  
*Stack Flexible Probabilities*

**Description**

This function mimics dplyr `bind`. It's useful if you have different ffp objects and want to stack them in the tidy (long) format.

**Usage**

`bind_probs(...)`

**Arguments**

`...`  
ffp objects to combine.

**Value**

A tidy tibble.

The output adds two new columns:

- `rowid` (an integer) with the row number of each realization;
- `key` (a factor) that keeps track of the ffp inputs as separated objects.

**See Also**

`crisp exp_decay kernel_normal kernel_entropy double_decay`

**Examples**

```r
library(ggplot2)
library(dplyr, warn.conflicts = FALSE)

x <- exp_decay(EuStockMarkets, 0.001)
y <- exp_decay(EuStockMarkets, 0.002)

bind_probs(x, y)
```
```r
bind_probs(x, y) %>%
ggplot(aes(x = rowid, y = probs, color = key)) +
geom_line() +
scale_color_viridis_d()
```

---

**bootstrap_scenarios**  
*Flexible Probabilities Driven Bootstrap*

**Description**
Resamples historical scenarios with flexible probabilities while keeping the empirical structure of the copulas intact.

**Usage**

```r
bootstrap_scenarios(x, p, n)
```

## S3 method for class 'numeric'

```r
bootstrap_scenarios(x, p, n)
```

## S3 method for class 'matrix'

```r
bootstrap_scenarios(x, p, n)
```

## S3 method for class 'ts'

```r
bootstrap_scenarios(x, p, n)
```

## S3 method for class 'xts'

```r
bootstrap_scenarios(x, p, n)
```

## S3 method for class 'tbl'

```r
bootstrap_scenarios(x, p, n)
```

## S3 method for class 'data.frame'

```r
bootstrap_scenarios(x, p, n)
```

**Arguments**

- `x`  
  A time series defining the scenario-probability distribution.

- `p`  
  An object of the `ffp` class.

- `n`  
  An integer scalar with the number of scenarios to be generated.

**Details**

The argument `x` is supposed to have the same size of `p`. 

**Value**

A tibble with the number of rows equal to \( n \).

**Examples**

```r
set.seed(123)
ret <- diff(log(EuStockMarkets))
ew <- rep(1 / nrow(ret), nrow(ret))

bootstrap_scenarios(x = ret, p = as_ffp(ew), n = 10)
```

---

**crisp**

*Full Information by Market Conditioning*

**Description**

Give full weight to occurrences when a macroeconomic statement satisfies a logical condition.

**Usage**

```r
crisp(x, lgl)
```

```r
## Default S3 method:
crisp(x, lgl)
## S3 method for class 'numeric'
crisp(x, lgl)
## S3 method for class 'matrix'
crisp(x, lgl)
## S3 method for class 'ts'
crisp(x, lgl)
## S3 method for class 'xts'
crisp(x, lgl)
## S3 method for class 'data.frame'
crisp(x, lgl)
## S3 method for class 'tbl_df'
crisp(x, lgl)
```

**Arguments**

- **x**
  
  An univariate or a multivariate distribution.

- **lgl**
  
  A logical vector with TRUE’s and FALSE’s indicating which scenarios should be considered.
Value

A numerical vector of class ffp with the new probabilities distribution.

See Also

exp_decay kernel_normal

Examples

library(ggplot2)
# invariance (stationarity)
ret <- diff(log(EuStockMarkets))

# full weight on scenarios where CAC operated above 2%
market_condition <- crisp(x = ret, ret[, 3] > 0.02)

autoplot(market_condition) +
  scale_color_viridis_c()
**db_tbl**

Dataset used in Historical Scenarios with Fully Flexible Probabilities (tibble format).

**Usage**

db_tbl

**Format**

An object of class tbl_df (inherits from tbl.data.frame) with 1083 rows and 9 columns.

**See Also**

db

double_decay

**Flexible Probabilities using Partial Information**

**Description**

This function uses entropy-pooling to match different decay-factors on the covariance matrix.

**Usage**

double_decay(x, slow, fast)

## Default S3 method:

double_decay(x, slow, fast)

## S3 method for class 'numeric'

double_decay(x, slow, fast)

## S3 method for class 'matrix'

double_decay(x, slow, fast)

## S3 method for class 'ts'

double_decay(x, slow, fast)

## S3 method for class 'xts'

double_decay(x, slow, fast)
## S3 method for class 'tbl'
double_decay(x, slow, fast)

## S3 method for class 'data.frame'
double_decay(x, slow, fast)

### Arguments
- **x**: An univariate or a multivariate distribution.
- **slow**: A number with the long half-life (slow decay) for the correlation matrix.
- **fast**: A number with the short-life (high decay) for the volatility.

### Value
A numerical vector of class ffp with the new probabilities distribution.

### References

### See Also
- `kernel_entropy`
- `half_life`

### Examples
```r
library(ggplot2)
slow <- 0.0055
fast <- 0.0166
ret <- diff(log(EuStockMarkets))

dd <- double_decay(ret, slow, fast)
dd

autoplot(dd) +
  scale_color_viridis_c()
```

---

### empirical_stats

**Summary Statistics for Empirical Distributions**

### Description
Computes the mean, standard deviation, skewness, kurtosis, Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR) under flexible probabilities.
Usage

```
empirical_stats(x, p, level = 0.01)
```

## Default S3 method:
```
empirical_stats(x, p, level = 0.01)
```

## S3 method for class 'numeric'
```
empirical_stats(x, p, level = 0.01)
```

## S3 method for class 'matrix'
```
empirical_stats(x, p, level = 0.01)
```

## S3 method for class 'xts'
```
empirical_stats(x, p, level = 0.01)
```

## S3 method for class 'ts'
```
empirical_stats(x, p, level = 0.01)
```

## S3 method for class 'data.frame'
```
empirical_stats(x, p, level = 0.01)
```

## S3 method for class 'tbl_df'
```
empirical_stats(x, p, level = 0.01)
```

Arguments

- `x` A time series defining the scenario-probability distribution.
- `p` An object of the `ffp` class.
- `level` A number with the desired probability level. The default is `level = 0.01`.

Details

The data in `x` and `p` are expected to have the same number of rows (size).

Value

A tidy tibble with 3 columns:

- stat: a column with Mu, Std, Skew, Kurt, VaR and CVaR.
- name: the asset names.
- value: the computed value for each statistic.

Examples

```
library(dplyr, warn.conflicts = FALSE)
library(ggplot2)

ret <- diff(log(EuStockMarkets))
```
# with equal weights (standard scenario)
ew <- rep(1 / nrow(ret), nrow(ret))
empirical_stats(x = ret, p = as_ffp(ew)) %>%
  ggplot(aes(x = name, y = value)) +
  geom_col() +
  facet_wrap(~stat, scales = "free") +
  labs(x = NULL, y = NULL)

# with ffp
exp_smooth <- exp_decay(ret, 0.015)
empirical_stats(ret, exp_smooth) %>%
  ggplot(aes(x = name, y = value)) +
  geom_col() +
  facet_wrap(~stat, scales = "free") +
  labs(x = NULL, y = NULL)

---

**ens**  

**Effective Number of Scenarios**

**Description**

Shows how many scenarios are effectively been considered when using flexible probabilities.

**Usage**

`ens(p)`

**Arguments**

- `p`  
  An object of the ffp class.

**Value**

A single double.

**Examples**

```r
set.seed(123)
p <- exp_decay(stats::rnorm(100), 0.01)

# ens is smaller than 100
ens(p)
```
**Description**

Exponential smoothing twists probabilities by giving relatively more weight to recent observations at an exponential rate.

**Usage**

```r
exp_decay(x, lambda)
```

- **Default S3 method:**
  ```r
exp_decay(x, lambda)
  ```

- **S3 method for class 'numeric'**
  ```r
exp_decay(x, lambda)
  ```

- **S3 method for class 'matrix'**
  ```r
exp_decay(x, lambda)
  ```

- **S3 method for class 'ts'**
  ```r
exp_decay(x, lambda)
  ```

- **S3 method for class 'xts'**
  ```r
exp_decay(x, lambda)
  ```

- **S3 method for class 'data.frame'**
  ```r
exp_decay(x, lambda)
  ```

- **S3 method for class 'tbl'**
  ```r
exp_decay(x, lambda)
  ```

**Arguments**

- **x** An univariate or a multivariate distribution.
- **lambda** A number for the decay parameter.

**Details**

The half-life is linked with the lambda parameter as follows:

- \( \text{HL} = \log(2) / \lambda \).

For example: \( \log(2) / 0.0166 \) is approximately 42. So, a parameter \( \lambda \) of 0.0166 can be associated with a half-life of two-months.
Value

A numerical vector of class ffp with the new probabilities distribution.

See Also

crisp kernel_normal half_life

Examples

library(ggplot2)

# long half_life
long_hl <- exp_decay(EuStockMarkets, 0.001)
long_hl
autoplot(long_hl) +
  scale_color_viridis_c()

# short half_life
short_hl <- exp_decay(EuStockMarkets, 0.015)
short_hl
autoplot(short_hl) +
  scale_color_viridis_c()
Arguments

x • For ffp(): A numeric vector.
   • For is_ffp(): An object to be tested.
   • For as_ffp(): An object to convert to ffp.
   ... Additional attributes to be passed to ffp.

Details

The ffp class is designed to interact with doubles, but the output of c(ffp, double) or c(double, ffp) will always return a double (not an ffp object), since there is no way to guarantee the interaction between a numeric vector and a probability will also be a probability.

Value

• ffp() and as_ffp() return an S3 vector of class ffp (built upon double’s);
• is_ffp() returns a logical object.

Examples

set.seed(123)
p <- runif(5)
p <- p / sum(p)
is_ffp(p)
ffp(p)


half_life

Half-Life Calculation

Description

Compute the implied half-life of a decay parameter.

Usage

half_life(lambda)

Arguments

lambda A number.

Value

A single number with the half-life in days.

See Also

exp_decay double_decay
Examples

half_life(0.0166)
half_life(0.01)

kernel_entropy

Partial Information Kernel-Damping

Description

This function uses entropy-pooling to find the probability distribution that can constrain the first two moments while imposing the minimal structure in the data.

Usage

kernel_entropy(x, mean, sigma = NULL)

## Default S3 method:  
kernel_entropy(x, mean, sigma = NULL)

## S3 method for class 'numeric'  
kernel_entropy(x, mean, sigma = NULL)

## S3 method for class 'matrix'  
kernel_entropy(x, mean, sigma = NULL)

## S3 method for class 'ts'  
kernel_entropy(x, mean, sigma = NULL)

## S3 method for class 'xts'  
kernel_entropy(x, mean, sigma = NULL)

## S3 method for class 'tbl_df'  
kernel_entropy(x, mean, sigma = NULL)

## S3 method for class 'data.frame'  
kernel_entropy(x, mean, sigma = NULL)

Arguments

x An univariate or a multivariate distribution.
mean A numeric vector in which the kernel should be centered.
sigma The uncertainty (volatility) around the mean. When NULL, only the mean is constrained.

Value

A numerical vector of class ffp with the new probabilities distribution.
kernel_normal

See Also

double_decay

Examples

library(ggplot2)

ret <- diff(log(EuStockMarkets[, 1]))
mean <- -0.01 # scenarios around -1%
sigma <- var(diff(ret))

ke <- kernel_entropy(ret, mean, sigma)
ke

autoplot(ke) +
  scale_color_viridis_c()

Full Information by Kernel-Damping

Description

In this framework, historical realizations receive a weight proportional to its distance from a target mean that is surrounded by normal kernel.

Usage

kernel_normal(x, mean, sigma)

## Default S3 method:
kernel_normal(x, mean, sigma)

## S3 method for class 'numeric'
kernel_normal(x, mean, sigma)

## S3 method for class 'matrix'
kernel_normal(x, mean, sigma)

## S3 method for class 'ts'
kernel_normal(x, mean, sigma)

## S3 method for class 'xts'
kernel_normal(x, mean, sigma)

## S3 method for class 'tbl_df'
kernel_normal(x, mean, sigma)

## S3 method for class 'data.frame'
kernel_normal(x, mean, sigma)
### scenario_density

**Plot Scenarios**

This function is designed to make it easier to visualize the impact of a `View` in the P&L distribution.

#### Usage

- `scenario_density(x, p, n = 10000)`
- `scenario_histogram(x, p, n = 10000)`

#### Arguments

- **x**: An univariate or a multivariate distribution.
- **mean**: A numeric vector in which the kernel should be centered.
- **sigma**: The uncertainty (volatility) around the mean.

#### Value

A numerical vector of class `ffp` with the new probabilities distribution.

#### See Also

- `crisp`
- `exp_decay`

#### Examples

```r
library(ggplot2)

ret <- diff(log(EuStockMarkets[, 1]))
mean <- -0.01 # scenarios around -1%
sigma <- var(diff(ret))

kn <- kernel_normal(ret, mean, sigma)
kn

autoplot(kn) +
  scale_color_viridis_c()

# A larger sigma spreads out the distribution
sigma <- var(diff(ret)) / 0.05
kn <- kernel_normal(ret, mean, sigma)

autoplot(kn) +
  scale_color_viridis_c()
```
**scenario_density**

**Arguments**

- `x` An univariate marginal distribution.
- `p` A probability from the `ffp` class.
- `n` An integer scalar with the number of scenarios to be generated.

**Details**

To generate a scenario-distribution the margins are bootstrapped using `bootstrap_scenarios`. The number of resamples can be controlled with the `n` argument (default is `n = 10000`).

**Value**

A `ggplot2` object.

**Examples**

```r
pnl <- diff(log(EuStockMarkets))[, 1]
p <- exp_decay(pnl, 0.005)

scenario_density(pnl, p, 500)
scenario_histogram(pnl, p, 500)
```
Index

* datasets
  - db, 6
  - db_tbl, 7

as_ffp(ffp), 12
autoplot.ffp, 2

bind, 3
bind_probs, 3
bootstrap_scenarios, 4, 17

crisp, 3, 5, 12, 16

db, 6, 7
db_tbl, 6, 7
double_decay, 3, 7, 13, 15

empirical_stats, 8
ens, 10
exp_decay, 3, 6, 11, 13, 16

ffp, 12

half_life, 8, 12, 13

is_ffp(ffp), 12

kernel_entropy, 3, 8, 14
kernel_normal, 3, 6, 12, 15

plot.ffp(autoplot.ffp), 2

scenario_density, 16
scenario_histogram(scenario_density), 16