Package: ffp (via r-universe)

September 18, 2024

Type Package

Title Fully Flexible Probabilities for Stress Testing and Portfolio Construction

Version 0.2.2.9000

Description Implements numerical entropy-pooling for portfolio construction and scenario analysis as described in Meucci, Attilio (2008) and Meucci, Attilio (2010) <doi:10.2139/ssrn.1696802>.

License MIT + file LICENSE

URL https://github.com/Reckziegel/FFP

BugReports https://github.com/Reckziegel/FFP/issues

Depends R (>= 2.10)

Imports assertthat (>= 0.2.1), dplyr (>= 1.0.10), forcats (>= 0.5.2), ggdist (>= 3.2.0), ggplot2 (>= 3.3.6), lubridate (>= 1.8.0), magrittr (>= 2.0.3), methods, mvtnorm (>= 1.1-3), purrr (>= 0.3.4), rlang (>= 1.0.6), scales (>= 1.2.1), stringr (>= 1.4.1), stats, tibble (>= 3.1.8), tidyr (>= 1.2.1), vctrs (>= 0.4.1), nloptr (>= 2.0.3), crayon, NlcOptim (>= 0.6)

Suggests copula, covr, ghyp, knitr (>= 1.40), markdown, rmarkdown, roxygen2, spelling, testthat (>= 3.1.4), xts (>= 0.12.1)

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

VignetteBuilder knitr

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

RemoteUrl https://github.com/reckziegel/ffp

RemoteRef HEAD

RemoteSha da655b211d72304d7bf436a5116fb3b4c40d5a0f

Contents

autoplot.ffp . . . 2
bind_probs . . . 3
bind_views . . . 4
bootstrap_scenarios . . . 5
crisp . . . 6
db . . . 7
db_tbl . . . 8
double_decay . . . 8
empirical_stats . . . 10
ens . . . 11
entropy_pooling . . . 12
exp_decay . . . 14
ffp . . . 15
ffp_moments . . . 16
half_life . . . 17
kernel_entropy . . . 18
kernel_normal . . . 19
relative_entropy . . . 20
scenario_density . . . 21
view_on_copula . . . 22
view_on_correlation . . . 23
view_on_joint_distribution . . . 24
view_on_marginal_distribution . . . 26
view_on_mean . . . 27
view_on_rank . . . 28
view_on_volatility . . . 30

Index **32**

`autoplot.ffp` *Inspection of a* `ffp` *object with ggplot2*

Description

Extends the `autoplot` method for the `ffp` class.

Usage

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

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

Arguments

object	An objected 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.

Examples

```
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

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

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

bind_probs(x, y)

bind_probs(x, y) %>%
  ggplot(aes(x = rowid, y = probs, color = fn)) +
  geom_line() +
  scale_color_viridis_d() +
  theme(legend.position="bottom")
```

---

bind_views *Stack Different Views*

Description

Bind views for entropy programming.

Usage

```
bind_views(...)
```

Arguments

...	Objects of the class ffp_views to combine.

Value

A list of the view class.

Examples

```
library(ggplot2)

# Invariant
ret <- diff(log(EuStockMarkets))
n   <- nrow(ret)

# Prior probabilities (usually equal weight scheme)
prior <- rep(1 / n, n)
```

```
# Prior belief for expected returns (here is 0% for each asset)
view_mean <- view_on_mean(x = ret, mean = rep(0, 4))

#' view on volatility
vol <- apply(ret, 2, stats::sd) * 1.1 # volatility 10% higher than average
view_volatility <- view_on_volatility(x = ret, vol = vol)

views_comb <- bind_views(view_mean, view_volatility)
views_comb

ep <- entropy_pooling(p      = prior,
                      Aeq    = views_comb$Aeq,
                      beq    = views_comb$beq,
                      A      = views_comb$A,
                      b      = views_comb$b,
                      solver = "nlminb")
autoplot(ep)
```

---

bootstrap_scenarios *Flexible Probabilities Driven Bootstrap*

Description

Resamples historical scenarios with flexible probabilities.

Usage

```
bootstrap_scenarios(x, p, n)

## S3 method for class 'numeric'
bootstrap_scenarios(x, p, n)

## S3 method for class 'matrix'
bootstrap_scenarios(x, p, n)

## S3 method for class 'ts'
bootstrap_scenarios(x, p, n)

## S3 method for class 'xts'
bootstrap_scenarios(x, p, n)

## S3 method for class 'tbl'
bootstrap_scenarios(x, p, n)

## S3 method for class 'data.frame'
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

```
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 that satisfies a logical condition.

Usage

```
crisp(x, lgl)

## 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 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 returns were above 2%
market_condition <- crisp(x = ret, ret[ , 3] > 0.02)
market_condition

autoplot(market_condition) +
  scale_color_viridis_c()
```

---

db *Dataset used in Historical Scenarios with Fully Flexible Probabilities (`matrix` format).*

Description

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

Usage

```
db
```

Format

An object of class `matrix` (inherits from `array`) with 1083 rows and 9 columns.

See Also

db_tbl

---

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

Description

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

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 double with the long half-life (slow decay) for the correlation matrix.
fast	A double with the short-life (high decay) for the volatility.

Value

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

References

De Santis, G., R. Litterman, A. Vesval, and K. Winkelmann, 2003, Covariance matrix estimation, Modern investment management: an equilibrium approach, Wiley.

See Also

kernel_entropy half_life

Examples

```
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

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

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

`entropy_pooling` *Numerical Entropy Minimization*

Description

This function solves the entropy minimization problem with equality and inequality constraints. The solution is a vector of posterior probabilities that distorts the least the prior (equal-weights probabilities) given the constraints (views on the market).

Usage

```
entropy_pooling(
  p,
  A = NULL,
  b = NULL,
  Aeq = NULL,
  beq = NULL,
  solver = c("nlminb", "solnl", "nloptr"),
  ...
)
```

Arguments

`p`	A vector of prior probabilities.
`A`	The linear inequality constraint (left-hand side).
`b`	The linear inequality constraint (right-hand side).
`Aeq`	The linear equality constraint (left-hand side).
`beq`	The linear equality constraint (right-hand side).
`solver`	A `character`. One of: "nlminb", "solnl" or "nloptr".
`...`	Further arguments passed to one of the solvers.

Details

When imposing views constraints there is no need to specify the non-negativity constraint for probabilities, which is done automatically by `entropy_pooling`.

For the arguments accepted in `...`, please see the documentation of `nlminb`, `solnl`, `nloptr` and the examples bellow.

Value

A vector of posterior probabilities.

Examples

```
# setup
ret <- diff(log(EuStockMarkets))
n   <- nrow(ret)

# View on expected returns (here is 2% for each asset)
mean <- rep(0.02, 4)

# Prior probabilities (usually equal weight scheme)
prior <- rep(1 / n, n)

# View
views <- view_on_mean(x = ret, mean = mean)

# Optimization
ep <- entropy_pooling(
 p      = prior,
 Aeq    = views$Aeq,
 beq    = views$beq,
 solver = "nlminb"
)
ep

### Using the ... argument to control the optimization parameters

# nlminb
ep <- entropy_pooling(
 p      = prior,
 Aeq    = views$Aeq,
 beq    = views$beq,
 solver = "nlminb",
 control = list(
     eval.max = 1000,
     iter.max = 1000,
     trace    = TRUE
   )
)
ep

# nloptr
ep <- entropy_pooling(
 p      = prior,
 Aeq    = views$Aeq,
 beq    = views$beq,
 solver = "nloptr",
 control = list(
     xtol_rel = 1e-10,
     maxeval  = 1000,
```

```
      check_derivatives = TRUE
    )
)
ep
```

---

exp_decay *Full Information by Exponential Decay*

Description

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

Usage

```
exp_decay(x, lambda)

## Default S3 method:
exp_decay(x, lambda)

## S3 method for class 'numeric'
exp_decay(x, lambda)

## S3 method for class 'matrix'
exp_decay(x, lambda)

## S3 method for class 'ts'
exp_decay(x, lambda)

## S3 method for class 'xts'
exp_decay(x, lambda)

## S3 method for class 'data.frame'
exp_decay(x, lambda)

## S3 method for class 'tbl'
exp_decay(x, lambda)
```

Arguments

x	An univariate or a multivariate distribution.
lambda	A double for the decay parameter.

Details

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

- 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 (21 * 2).

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()
```

`ffp` *Manipulate the* `ffp` *Class*

Description

Helpers and Constructors from `ffp`.

Usage

```
ffp(x = double(), ...)

is_ffp(x)

as_ffp(x)

## Default S3 method:
as_ffp(x)

## S3 method for class 'integer'
as_ffp(x)
```

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)
as_ffp(p)
```

---

ffp_moments *Moments with Flexible Probabilities*

Description

Computes the location and dispersion statistics under flexible probabilities.

Usage

```
ffp_moments(x, p = NULL)

## Default S3 method:
ffp_moments(x, p = NULL)

## S3 method for class 'numeric'
ffp_moments(x, p = NULL)

## S3 method for class 'matrix'
ffp_moments(x, p = NULL)

## S3 method for class 'xts'
```

```
ffp_moments(x, p = NULL)

## S3 method for class 'data.frame'
ffp_moments(x, p = NULL)

## S3 method for class 'tbl_df'
ffp_moments(x, p = NULL)
```

Arguments

x	A tabular (non-tidy) data structure.
p	An object of the ffp class.

Value

A list with 2 elements: mu and sigma.

Examples

```
x <- matrix(diff(log(EuStockMarkets)), ncol = 4)
colnames(x) <- colnames(EuStockMarkets)
p <- stats::runif(nrow(x))
p <- p / sum(p)

ffp_moments(x = x, p = p)

# compare with the standard approach
colMeans(x)
cov(x)
```

---

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

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.

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()
```

---

kernel_normal *Full Information by Kernel-Damping*

Description

Historical realizations receive a weight proportional to their distance from a target mean.

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)
```

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

```
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()
```

---

relative_entropy *Relative Entropy*

Description

Computes the relative entropy of two distributions.

Usage

```
relative_entropy(prior, posterior)
```

Arguments

prior	A prior probability distribution.
posterior	A posterior probability distribution.

Value

A double with the relative entropy.

Examples

```
set.seed(222)

prior <- rep(1 / 100, 100)

posterior <- runif(100)
posterior <- posterior / sum(posterior)

relative_entropy(prior, posterior)
```

---

scenario_density *Plot Scenarios*

Description

This functions are 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 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

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

scenario_density(x, p, 500)

scenario_histogram(x, p, 500)
```

---

view_on_copula *Views on Copulas*

Description

Helper to construct constraints on copulas for entropy programming.

Usage

```
view_on_copula(x, simul, p)

## Default S3 method:
view_on_copula(x, simul, p)

## S3 method for class 'matrix'
view_on_copula(x, simul, p)

## S3 method for class 'xts'
view_on_copula(x, simul, p)

## S3 method for class 'tbl_df'
view_on_copula(x, simul, p)
```

Arguments

x	A multivariate copula.
simul	A simulated target copula.
p	An object of the ffp class.

Value

A list of the view class.

Examples

```
set.seed(1)
library(ggplot2)

# Invariants
ret <- diff(log(EuStockMarkets))
u <- apply(ret, 2, stats::pnorm) # assuming normal copula
n <- nrow(u)

#' Prior probability distribution
prior <- rep(1 / n, n)

# Simulated marginals
simul_marg <- bootstrap_scenarios(ret, as_ffp(prior), as.double(n))

# Copulas derived from the simulated margins
simul_cop <- apply(simul_marg, 2, stats::pnorm) # assuming normal copula

views <- view_on_copula(x = u, simul = simul_cop, p = prior)
views

ep <- entropy_pooling(p = prior, Aeq = views$Aeq, beq = views$beq, solver = "nloptr")
autoplot(ep)
```

---

view_on_correlation *Views on Correlation Structure*

Description

Helper to construct views on the correlation matrix.

Usage

```
view_on_correlation(x, cor)

## Default S3 method:
view_on_correlation(x, cor)

## S3 method for class 'matrix'
view_on_correlation(x, cor)

## S3 method for class 'xts'
view_on_correlation(x, cor)

## S3 method for class 'tbl_df'
view_on_correlation(x, cor)
```

Arguments

x	An univariate or a multivariate distribution.
cor	A matrix for the target correlation structure of the series in x.

Value

A list of the view class.

Examples

```
library(ggplot2)

# Invariant
ret <- diff(log(EuStockMarkets))

# Assume that a panic event throws all correlations to the roof!
co <- matrix(0.95, 4, 4)
diag(co) <- 1
co

# Prior probability (usually the equal-weight setting)
prior <- rep(1 / nrow(ret), nrow(ret))

# View
views <- view_on_correlation(x = ret, cor = co)
views

# Optimization
ep <- entropy_pooling(p = prior, Aeq = views$Aeq, beq = views$beq, solver = "nlminb")
autoplot(ep)

# prior correlation structure
stats::cor(ret)

# posterior correlation structure matches the initial view very closely
stats::cov2cor(ffp_moments(x = ret, p = ep)$sigma)
```

---

view_on_joint_distribution

Views on Joint Distribution

Description

Helper to construct constraints on the entire distribution.

Usage

```
view_on_joint_distribution(x, simul, p)

## Default S3 method:
view_on_joint_distribution(x, simul, p)

## S3 method for class 'matrix'
view_on_joint_distribution(x, simul, p)

## S3 method for class 'xts'
view_on_joint_distribution(x, simul, p)

## S3 method for class 'tbl_df'
view_on_joint_distribution(x, simul, p)
```

Arguments

x	An univariate or a multivariate distribution.
simul	An univariate or multivariate simulated panel.
p	An object of the ffp class.

Details

- simul must have the same number of columns than x
- p should have the same number of rows that simul.

Value

A list of the view class.

Examples

```
set.seed(1)
library(ggplot2)

# Invariants
ret <- diff(log(EuStockMarkets))
n <- nrow(ret)

#' Prior probability distribution
prior <- rep(1 / n, n)

# Simulated marginals
simul <- bootstrap_scenarios(ret, as_ffp(prior), as.double(n))

views <- view_on_joint_distribution(x = ret, simul = simul, p = prior)
views

ep <- entropy_pooling(p = prior, Aeq = views$Aeq, beq = views$beq, solver = "nlminb")
```

```
autoplot(ep)

# location matches
colMeans(simul)
ffp_moments(x = ret, p = ep)$mu

# dispersion matches
cov(simul)
ffp_moments(x = ret, p = ep)$sigma
```

`view_on_marginal_distribution` *Views on Marginal Distribution*

Description

Helper to construct constraints on the marginal distribution.

Usage

```
view_on_marginal_distribution(x, simul, p)

## Default S3 method:
view_on_marginal_distribution(x, simul, p)

## S3 method for class 'matrix'
view_on_marginal_distribution(x, simul, p)

## S3 method for class 'xts'
view_on_marginal_distribution(x, simul, p)

## S3 method for class 'tbl_df'
view_on_marginal_distribution(x, simul, p)
```

Arguments

`x`	An univariate or a multivariate distribution.
`simul`	An univariate or multivariate simulated panel.
`p`	An object of the `ffp` class.

Details

- `simul` must have the same number of columns than `x`
- `p` should have the same number of rows that `simul`.

Value

A `list` of the `view` class.

Examples

```
set.seed(1)
library(ggplot2)

# Invariants
ret <- diff(log(EuStockMarkets))
n <- nrow(ret)

#' Prior probability distribution
prior <- rep(1 / n, n)

# Simulated marginals
simul <- bootstrap_scenarios(ret, as_ffp(prior), as.double(n))

views <- view_on_marginal_distribution(x = ret, simul = simul, p = prior)
views

ep <- entropy_pooling(p = prior, Aeq = views$Aeq, beq = views$beq, solver = "nlminb")
autoplot(ep)

# location matches
colMeans(simul)
ffp_moments(x = ret, p = ep)$mu

# dispersion matches
cov(simul)
ffp_moments(x = ret, p = ep)$sigma
```

`view_on_mean` *Views on Expected Returns*

Description

Helper to construct views on expected returns.

Usage

```
view_on_mean(x, mean)

## Default S3 method:
view_on_mean(x, mean)

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

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

## S3 method for class 'tbl_df'
view_on_mean(x, mean)
```

Arguments

x	An univariate or a multivariate distribution.
mean	A double for the target location parameter of the series in x.

Value

A list of the view class.

Examples

```
library(ggplot2)

# Invariant
ret <- diff(log(EuStockMarkets))
n   <- nrow(ret)

# View on expected returns (here is 2% for each asset)
mean <- rep(0.02, 4)

# Prior probabilities (usually equal weight scheme)
prior <- rep(1 / n, n)

# View
views <- view_on_mean(x = ret, mean = mean)
views

# Optimization
ep <- entropy_pooling(p = prior, Aeq = views$Aeq, beq = views$beq, solver = "nlminb")
autoplot(ep)

# Probabilities are twisted in such a way that the posterior
# `mu` match's exactly with previously stated beliefs
ffp_moments(x = ret, p = ep)$mu
```

---

view_on_rank *Views on Relative Performance*

Description

Helper to construct views on relative performance of assets.

Usage

```
view_on_rank(x, rank)

## Default S3 method:
view_on_rank(x, rank)
```

```
## S3 method for class 'matrix'
view_on_rank(x, rank)

## S3 method for class 'xts'
view_on_rank(x, rank)

## S3 method for class 'tbl_df'
view_on_rank(x, rank)
```

Arguments

x	An univariate or a multivariate distribution.
rank	A integer with the assets rank (from the worst to the best performer).

Details

If rank = c(2, 1) it is implied that asset in the first column will outperform the asset in the second column. For longer vectors the interpretation is the same: assets on the right will outperform assets on the left.

Value

A list of the view class.

Examples

```
library(ggplot2)

# Invariants
x <- diff(log(EuStockMarkets))
prior <- rep(1 / nrow(x), nrow(x))

# asset in the first col will outperform the asset in the second col (DAX will
# outperform SMI).
views <- view_on_rank(x = x, rank = c(2, 1))
views

ep <- entropy_pooling(p = prior, A = views$A, b = views$b, solver = "nloptr")
autoplot(ep)

# Prior Returns (SMI > DAX)
colMeans(x)[1:2]

# Posterior Returns (DAX > SMI)
ffp_moments(x, ep)$mu[1:2]
```

`view_on_volatility` *Views on Volatility*

Description

Helper to construct views on volatility.

Usage

```
view_on_volatility(x, vol)

## Default S3 method:
view_on_volatility(x, vol)

## S3 method for class 'matrix'
view_on_volatility(x, vol)

## S3 method for class 'xts'
view_on_volatility(x, vol)

## S3 method for class 'tbl_df'
view_on_volatility(x, vol)
```

Arguments

`x`	An univariate or a multivariate distribution.
`vol`	A `double` for the target volatility structure of the series in `x`.

Value

A `list` of the `view` class.

Examples

```
library(ggplot2)

# Invariant
ret <- diff(log(EuStockMarkets))
n   <- nrow(ret)

# Expected a volatility 30% higher than historical average
vol <- apply(ret, 2, stats::sd) * 1.3

# Prior Probabilities
prior <- rep(1 / n, n)

# Views
views <- view_on_volatility(x = ret, vol = vol)
```

```
views

# Optimization
ep <- entropy_pooling(p = prior, Aeq = views$Aeq, beq = views$beq, solver = "nlminb")
autoplot(ep)

# Desired volatility
vol

# Posterior volatility matches very closely with the desired volatility
sqrt(diag(ffp_moments(x = ret, p = ep)$sigma))
```

Index

∗ **datasets**
    db, 7
    db_tbl, 8

as_ffp *(*ffp*)*, 15
autoplot.ffp, 2

bind, *3*
bind_probs, 3
bind_views, 4
bootstrap_scenarios, 5, *21*

crisp, *4*, 6, *15*, *20*

db, 7, *8*
db_tbl, *8*, 8
double_decay, *4*, 8, *18, 19*

empirical_stats, 10
ens, 11
entropy_pooling, 12
exp_decay, *4*, *7*, 14, *18*, *20*

ffp, 15
ffp_moments, 16

half_life, *9*, *15*, 17

is_ffp *(*ffp*)*, 15

kernel_entropy, *4*, *9*, 18
kernel_normal, *4*, *7*, *15*, 19

nlminb, *12*
nloptr, *12*

plot.ffp *(*autoplot.ffp*)*, 2

relative_entropy, 20

scenario_density, 21
scenario_histogram *(*scenario_density*)*, 21
solnl, *12*

view_on_copula, 22
view_on_correlation, 23
view_on_joint_distribution, 24
view_on_marginal_distribution, 26
view_on_mean, 27
view_on_rank, 28
view_on_volatility, 30