Package: INFOSET (via r-universe)

September 8, 2024

Type Package

Title Computing a New Informative Distribution Set of Asset Returns

Version 4.0.6

Author Gloria Polinesi [aut, cre], Francesca Mariani [aut], Maria Cristina Recchioni [aut]

Maintainer Gloria Polinesi <g.polinesi@staff.univpm.it>

Description Estimation of the most-left informative set of gross returns (i.e., the informative set). The procedure to compute the informative set adjusts the method proposed by Mariani et al. (2022a) <doi:10.1007/s11205-020-02440-6> and Mariani et al. (2022b) <doi:10.1007/s10287-022-00422-2> to gross returns of financial assets. This is accomplished through an adaptive algorithm that identifies sub-groups of gross returns in each iteration by approximating their distribution with a sequence of two-component log-normal mixtures. These sub-groups emerge when a significant change in the distribution occurs below the median of the financial returns, with their boundary termed as the “change point" of the mixture. The process concludes when no further change points are detected. The outcome encompasses parameters of the leftmost mixture distributions and change points of the analyzed financial time series.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports Matrix, colorspace, dendextend, quadprog, mixtools, stats, graphics

Depends R (>= 2.10)

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2024-09-06 13:00:09 UTC

Contents

`asset.label` *Data for clustering and labeling ETFs*

Description

Contains asset class of ETFs

Usage

```
asset.label
```

Format

A data frame with 44 observations (rows) on 3 variables (columns)

id name of ETF

label from 1 to 5 according to the specific asset class

class specific asset class (5 categories)

Source

Created in-house to serve as an example

Examples

```
data(asset.label)
```

g_ret *Function to compute gross returns.*

Description

Calculate gross returns from prices.

Usage

```
g_ret(x)
```

Arguments

x	data object containing ordered price observations

Value

An object of the same class as x with the gross returns

infoset *Procedure to find the most-left distribution set.*

Description

Estimation of the vector of unknown parameters for the density functions associated with the two mixture components.

Usage

```
infoset(y)
```

Arguments

y	object of class "g_ret"

Value

An object of class "infoset" is a list containing the following components for the firse two iterations (k=2):

change.points a vector of change points.

prior.probability the a priori probabilities.

first.type.errors the cumulative distribution functions associated with the leftmost component of the mixture.

second.type.errors the cumulative distribution functions associated with the rightmost component of the mixture.

mean the parameters (drift) of the left-hand component of the log-normal mixture.

sd the parameters (volatility) of the left-hand component of the log-normal mixture.

References

Mariani, F., Polinesi, G., Recchioni, M. C. (2022). A tail-revisited Markowitz mean-variance approach and a portfolio network centrality. Computational Management Science, 19(3), 425-455.

Mariani, F., Ciommi, M., Chelli, F. M., Recchioni, M. C. (2020). An iterative approach to stratification: Poverty at regional level in Italy. Social Indicators Research, 1-31.

Examples

```
gross.ret<-as.data.frame(lapply(sample.data, g_ret))
infoset(gross.ret$ETF_1)


result<-NULL
for(i in 1:ncol(gross.ret)){
result[[i]]<-infoset(gross.ret[,i])
}
output<-matrix(unlist(result),12,ncol=ncol(gross.ret)) # output contains the information set
output<-t(output)
rownames(output)<-colnames(gross.ret)
colnames(output)<-c("ch_1","ch_2","priori_1","priori_2","first_1",
                "first_2","second_1","second_2","mean_1","mean_2","dev_1", "dev_2")
output<- as.data.frame(output)



############################################################
## EXAMPLE 1: Clustering ETFs
############################################################


library(colorspace)
library (dendextend)
group_label <- as.factor(asset.label$label)
d <- dist(output, method = 'euclidean')
hc_SIMS <- hclust(d, method = 'complete')
dend_SIMS <- as.dendrogram(hc_SIMS)
dend_SIMS <- color_branches(dend_SIMS, k = 4, col = c(1:4))
labels_colors(dend_SIMS) <-
     rainbow_hcl(5)[sort_levels_values(as.numeric(group_label)[order.dendrogram(dend_SIMS)])]
labels(dend_SIMS) <- paste(as.character(group_label)[order.dendrogram(dend_SIMS)],
       '(', labels(dend_SIMS), ')', sep = '')
dend_SIMS <- hang.dendrogram(dend_SIMS, hang_height = 0.001)
dend_SIMS <- assign_values_to_leaves_nodePar(dend_SIMS, 0.5, 'lab.cex')
dev.new()
old_par <- par(no.readonly = TRUE)
on.exit(par(old_par))
par(mar = c(1.8, 1.8, 1.8, 1))
plot(dend_SIMS, main = 'Complete linkage (the labels give the true ETF class)',
     horiz = TRUE, nodePar = list(cex = 0.007))
legend('topleft', legend = c('emerging equity Asia', 'emerging equity America',
                             'corporate bond', 'commodities', 'aggregate bond'),
```

```
          fill = c('#BDAB66', '#65BC8C', '#C29DDE', '#E495A5', '#55B8D0'), border = 'white')


############################################################
## EXAMPLE 2: Labelling ETFs
############################################################

ch <- output$ch_1
h <- list()
diff <- list()
n <- list()
a_int <- list()
hh <- list()
hh_p <- list()
expected_loss <- list()
p_loss<-list()
LR_cp <- list()
for(i in 1:ncol(gross.ret)) {
  h[[i]] <- hist(log(gross.ret[[i]]))
  diff[[i]] <- h[[i]]$breaks - ch[[i]]
  n[[i]] <- length(diff[[i]][diff[[i]]<0])
  a_int[[i]] <- h[[i]]$breaks[2] - h[[i]]$breaks[1]
  hh[[i]] <- h[[i]]$mids*h[[i]]$density*a_int[[i]]
  hh_p[[i]] <- h[[i]]$density*a_int[[i]]
  expected_loss[[i]] <- sum(hh[[i]][1:(n[[i]] - 1)]) +
    h[[i]]$density[n[[i]]]*(ch[[i]]-
                            h[[i]]$breaks[n[[i]]])*0.5*(ch[[i]] + h[[i]]$breaks[n[[i]]])
  p_loss[[i]]<-sum(hh_p[[i]][1:(n[[i]] - 1)]) +
    h[[i]]$density[n[[i]]]*(ch[[i]] - h[[i]]$breaks[n[[i]]])
  LR_cp[[i]] <- - (expected_loss[[i]]/p_loss[[i]])
}

############################################################
## EXAMPLE 3: Portfolio construction
############################################################


W <- list()
for(t in 0:15){
  W[[(t+1)]] = sample.data[(1 + t*125):(1290 + t*125), ]
}
ret <- list()
y <- list()
for(i in 1:ncol(sample.data)){
  for (t in 1:length(W)){
    ret[[t]] <- matrix(0,nrow = nrow(W[[1]]) - 1, ncol = ncol(sample.data))
    y[[t]] <- matrix(0, nrow = nrow(W[[1]]) - 1, ncol = ncol(sample.data))
  }
}

ch <- log(output$ch_1)
h <- list()
diff <- list()
```

```
n <- list()
a_int <- list()
hh <- list()
LR_cp <- list()
hh_p <- list()
p_loss <- list()
expected_loss <- list()

i <- ncol(sample.data)
for(t in 1:length(W)){
  h[[t]] <- vector('list', i)
  diff[[t]] <- vector('list', i)
  n[[t]] <- vector('list', i)
  a_int[[t]] <- vector('list', i)
  hh[[t]] <- vector('list', i)
  LR_cp[[t]]<- vector('list', i)
  hh_p[[t]] <- vector('list', i)
  p_loss[[t]] <- vector('list', i)
  expected_loss[[t]] <- vector('list', i)
}

for(i in 1:ncol(sample.data)){
  for (t in 1:length(W)){
    ret[[t]][,i] <- diff(log(W[[t]][, i]))
    y[[t]][,i] <- exp(ret[[t]][, i])
    y[[t]][,i] <- sort(y[[t]][, i]) ##gross return
    h[[t]][[i]] <- hist(log(y[[t]][, i]))
    diff[[t]][[i]] <- h[[t]][[i]]$breaks-ch[[i]]
    n[[t]][[i]] <- length(diff[[t]][[i]][diff[[t]][[i]] < 0])
    a_int[[t]][[i]] <- h[[t]][[i]]$breaks[2] - h[[t]][[i]]$breaks[1]
    hh[[t]][[i]] <- h[[t]][[i]]$mids*h[[t]][[i]]$density*a_int[[t]][[i]]
    hh_p[[t]][[i]] <- h[[t]][[i]]$density*a_int[[t]][[i]]
    expected_loss[[t]][[i]] <- sum(hh[[t]][[i]][1:(n[[t]][[i]] - 1)]) +
    h[[t]][[i]]$density[n[[t]][[i]]]*(ch[[i]] - h[[t]][[i]]$breaks[n[[t]][[i]]])*0.5*
   (ch[[i]] + h[[t]][[i]]$breaks[n[[t]][[i]]])
    p_loss[[t]][[i]] <- sum(hh_p[[t]][[i]][1:(n[[t]][[i]] - 1)]) +
    h[[t]][[i]]$density[n[[t]][[i]]]*(ch[[i]] - h[[t]][[i]]$breaks[n[[t]][[i]]])
    LR_cp[[t]][[i]] <- -(expected_loss[[t]][[i]]/p_loss[[t]][[i]])
  }
}

# ## Markowitz portfolio
library(quadprog)
library(Matrix)
r <- list()
meanret <- list()
COV_ret <- list()

for(t in 1: length(W)){
  r[[t]] <- matrix(colMeans(ret[[t]]))
  meanret[[t]] <- sum(r[[t]])/ncol(sample.data)
  COV_ret[[t]] <- cov(ret[[t]][(1289 - 252):1289, ])
  COV_ret[[t]] <- nearPD(COV_ret[[t]])$mat
```

```
}

tw <- length(W) - 1
n <- ncol(sample.data)
B <- list()
f <- list()
sol <- list()
w_ret <- list()
Pvalue_ret <- NULL
Pvalue_M <- list()
For6m_ret1 <- NULL
for(t in 1:tw){
  B[[t]] <- matrix(1, 1, n)
  B[[t]] <- rbind(B[[t]], t(r[[t]]), diag(n), -diag(n))
  f[[t]] <- c(1, meanret[[t]], rep(0, n), rep(-1, n))
  sol[[t]] <- solve.QP(Dmat = COV_ret[[t]], dvec = 0*r[[t]], Amat = t(B[[t]]),
        bvec = f[[t]], meq = 2)
  w_ret[[t]] <- round(sol[[t]]$solution,6)
  Pvalue_ret[t] <- sum(w_ret[[t]]*W[[t]][1289, ])
  Pvalue_M[[t]] <- (t(w_ret[[t]])%*%t(W[[t+1]][(1289 - 125):1289,]) -
                 Pvalue_ret[t])/Pvalue_ret[t]
  For6m_ret1[t] <- Pvalue_M[[t]][60]
}

# ## Combined Markowitz

for(t in 1:length(W)){
  LR_cp[[t]] <- matrix(unlist(LR_cp[[t]]), nrow = 1, ncol = ncol(sample.data))
}
B <- list()
f <- list()
sol <- list()
w_ret <- list()
Pvalue_ret <- NULL
Pvalue_C <- list()
For6m_ret2 <- NULL
lambda <- 0.0001
for(t in 1: tw){
  B[[t]] <- matrix(1, 1, n)
  B[[t]] <- rbind(B[[t]], t(r[[t]]), diag(n), -diag(n))
  f[[t]] <- c(1, meanret[[t]], rep(0, n), rep(-1, n))
  sol[[t]] <- solve.QP(Dmat = COV_ret[[t]], dvec = -lambda*LR_cp[[t]], Amat = t(B[[t]]),
                       bvec = f[[t]], meq = 2)
  w_ret[[t]] <- round(sol[[t]]$solution, 6)
  Pvalue_ret[t] <- sum(w_ret[[t]]*W[[t]][1289, ])
  Pvalue_C[[t]] <- (t(w_ret[[t]])%*%t(W[[t+1]][(1289-125):1289,]) -
                Pvalue_ret[t])/Pvalue_ret[t]
  For6m_ret2[t] <- Pvalue_C[[t]][60]
}

# ## Classical EDC
ret <- list()
y <- list()
```

```
for(i in 1:ncol(sample.data)){
  for(t in 1:length(W)){
    ret[[t]] <- matrix(0, nrow = nrow(W[[1]]) - 1, ncol = ncol(sample.data))
    y[[t]] <- matrix(0, nrow = nrow(W[[1]]) - 1, ncol = ncol(sample.data))
  }
}
for(i in 1:ncol(sample.data)){
  for (t in 1:length(W)){
    ret[[t]][, i] <- diff(log(W[[t]][, i]))
    y[[t]][, i] <- exp(ret[[t]][, i])
  }
}

W_out <- list()
for(t in 1:16){
  W_out[[(t)]] = ret[[t]][(1289 - 252):1289, ]
}
quant <- matrix(0, nrow = length(W), ncol = ncol(sample.data))
diff <- list()
for(t in 1:length(W)){
  for(i in 1:ncol(sample.data)){
    quant[t, i] <-quantile(as.numeric(W_out[[(t)]][, i]), probs = 0.05)
    diff[[t]] <- W_out[[(t)]] - colMeans(W_out[[(t)]])
    for (j in 1:length(W_out[[t]][, i])){
      if(W_out[[(t)]][j, i] > quant[t, i]){diff[[t]][j, i] = 0}
    }
  }
}

C_edc <- list()

for(t in 1:length(W)){
  aux <- matrix(0, nrow = ncol(sample.data), ncol = ncol(sample.data))
  for(i in 1:ncol(sample.data)){
    for(j in 1:ncol(sample.data)){
      aux[i, j] <- (mean(diff[[t]][, i]*diff[[t]][, j]))
    }
  }
  C_edc[[t]] <- aux
}

r <- list()
meanret <- list()
stdev <- list()
COV_edc <- list()

for(t in 1:length(W_out)){
  r[[t]] <- matrix(colMeans(ret[[t]]))
  meanret[[t]] <- sum(r[[t]])/n
  stdev[[t]] <- apply(W_out[[t]], 2, sd)
  stdev[[t]] <- matrix(stdev[[t]])
  COV_edc[[t]] <- nearPD(C_edc[[t]])$mat
}
```

```
B <- list()
f <- list()
sol <- list()
w_edc <- list()
Pvalue_edc <- NULL
Pvalue_EDC <- list()
For6m_edc1 <- NULL
for(t in 1: 15){
  B[[t]]<- matrix(1,1,n)
  B[[t]]<- rbind(B[[t]], t(r[[t]]), diag(n),-diag(n))
  f[[t]] <- c(1, meanret[[t]], rep(0, n),rep(-1, n))
  sol[[t]] <- solve.QP(Dmat = COV_edc[[t]], dvec = 0*r[[t]],
           Amat = t(B[[t]]), bvec = f[[t]], meq = 2)
  w_edc[[t]]<-round(sol[[t]]$solution, 6)
  Pvalue_edc[t]=sum(w_edc[[t]]*W[[t]][1289, ])
  Pvalue_EDC[[t]] = (t(w_edc[[t]])%*%t(W[[t+1]][(1289-125):1289, ]) -
              Pvalue_edc[t])/Pvalue_edc[t]
  For6m_edc1[t] <- Pvalue_EDC[[t]][60]
}

## Combined EDC
B <- list()
f <- list()
sol <- list()
w_edc <- list()
Pvalue_edc <- NULL
Pvalue_mod_EDC <- list()
For6m_mod_edc1 <- NULL

for(t in 1: 15){
  B[[t]]<- matrix(1,1,n)
  B[[t]]<- rbind(B[[t]], t(r[[t]]), diag(n),-diag(n))
  f[[t]] <- c(1, meanret[[t]], rep(0, n),rep(-1, n))
  sol[[t]] <- solve.QP(Dmat = COV_edc[[t]], dvec = lambda*LR_cp[[t]],
             Amat = t(B[[t]]), bvec = f[[t]], meq = 2)
  w_edc[[t]]<-round(sol[[t]]$solution, 6)
  Pvalue_edc[t]=sum(w_edc[[t]]*W[[t]][1289, ])
  Pvalue_mod_EDC[[t]] = (t(w_edc[[t]])%*%t(W[[t+1]][(1289-125):1289, ]) -
            Pvalue_edc[t])/Pvalue_edc[t]
  For6m_mod_edc1[t] <- Pvalue_mod_EDC[[t]][60]
}

sample_M <- NULL ## Classical Markowitz
sample_C <- NULL ## Combined Markowitz
sample_EDC <- NULL ## Classical EDC
sample_mod_EDC <- NULL ## Combined EDC
icont <- 0
count <- 1:15
for (t in count){
  for(j in 1:125){
    icont = icont + 1;
    sample_M[icont] = Pvalue_M[[t]][1, j]
    sample_C[icont] = Pvalue_C[[t]][1, j]
```

```
    sample_EDC[icont] = Pvalue_EDC[[t]][1, j]
    sample_mod_EDC[icont] = Pvalue_mod_EDC[[t]][1, j]
  }
}
dev.new()
old_par <- par(no.readonly = TRUE)
on.exit(par(old_par))
par(mfrow = c(1, 4))
boxplot(sample_M, ylim = c(-0.20, 0.15), outline = FALSE,
       main = expression(paste('Classical Markowitz')))
abline(h = mean(sample_M), col = 'red')
boxplot(sample_C, ylim = c(-0.20,0.15), outline = FALSE,
       main = expression(paste('Combined Markowitz')))
abline(h = mean(sample_C), col = 'red')
boxplot(sample_EDC, ylim = c(-0.20, 0.15), outline = FALSE,
       main = expression(paste('Classical EDC')))
abline(h = mean(sample_EDC), col = 'red')
boxplot(sample_mod_EDC, ylim = c(-0.20, 0.15), outline = FALSE,
       main = expression(paste('Combined EDC')))
abline(h = mean(sample_EDC), col = 'red')
dev.new()
par(mfrow = c(2, 1))
date<-as.Date(sample.data.ts$Date,format='%m/%d/%Y')
date_parz = seq(from = 1291, to = 3165, by = 64)
m = length(date_parz)
date_parz[m] = 3165
date_1 <- date[1291:3165]
date_2 <- date[date_parz]
matplot(date_1, cbind(sample_M, sample_C), type = 'l',
        col = c('red', 'black'), lty = c(2, 3),
       ylab = 'profit & loss', xlab = '',xaxt='n', ylim = c(-0.20, 0.15), cex.lab = 1.2)
axis(1, date_2, format(date_2, '%m/%Y'), cex.axis = .9, las = 2)
legend('bottomright', legend = c('Classical Markowitz', 'Combined Markowitz'),
       col=c('red','black'),lty = c(2, 3))
matplot(date_1, cbind(sample_EDC, sample_mod_EDC), type = 'l',
        col = c('brown', 'blue'), lty = c(3,1),
       ylab = 'profit & loss', xlab = '', xaxt = 'n', ylim = c(-0.20, 0.15), cex.lab = 1.2)
axis(1, date_2, format(date_2, '%m/%Y'), cex.axis = .9, las = 2)
legend('bottomright', legend = c('Classical EDC', 'Combined EDC'),
      col = c('brown','blue'), lty = c(3, 1))
```

`sample.data` *Data for infoset function*

Description

Contains daily prices of ETFs

Usage

```
sample.data
```

Format

A data frame with 3174 rows and 44 columns

Source

Created in-house to serve as an example

Examples

```
data(sample.data)
```

`sample.data.ts` *Data with time points for portfolio construction using the LR_cp measure*

Description

Contains daily prices of ETFs

Usage

```
sample.data.ts
```

Format

A data frame with 3175 rows and 45 columns

Source

Created in-house to serve as an example

Examples

```
data(sample.data.ts)
```

`tail_mixture` *Function to find the most-left distribution set.*

Description

An adaptive clustering algorithm identifies sub-groups of gross returns at each iteration by approximating their distribution with a sequence of two-component log-normal mixtures.

Usage

```
tail_mixture(y, shift, n_it)
```

Arguments

`y`	vector or data frame
`shift`	double
`n_it`	integer

Value

data object

Index