Package 'FPMD'

May 9, 2021

Type Package

Title On Functional Processes with Multiple Discontinuities

Version 0.1.0

Author Jialiang Li, Yaguang Li and Tailen Hsing

Maintainer Yaguang Li <liyg@ustc.edu.cn>

Description This package handles the problem of estimating multiple discontinuities for a functional data process. A half-kernel approach is considered that addresses the inference of the total number, locations, and jump sizes of the changes.

License GPL (>= 2)

Depends R (>= 4.0.0)

Encoding UTF-8

LazyData true

Imports Rcpp (>= 1.0.4), RcppEigen, fdapace, zoo, locpol, pracma, extraDistr

LinkingTo Rcpp, RcppEigen

RoxygenNote 7.1.1

R topics documented:

Australian *Sydney Temperature Data.*

Description

Australian daily minimum temperature climate data form year 1855 to 2012 for Sydney (Observatory Hill) station.

Usage

```
data(Australian)
```

Format

dataSS Data matrix.

Lt Observation time points.

Ly Daily minimum temperature. ...

Source

http://www.bom.gov.au/climate/data

Examples

```
## Not run:
data(Australian)
dataSS = Australian$dataSS
Ly = Australian$Ly
Lt = Australian$Lt

## global test
bw.seq = seq(0.01, 0.11, by = 0.02)
reject = rep(NA, length(bw.seq))
## reject H0: no jump points
for (i in 1:length(bw.seq)) {
  reject[i] = MDTest(bw = bw.seq[i], alpha = 0.05, Lt = Lt, Ly = Ly, Wtype = 'MIX')
  cat('reject:', reject[i], 'for h = ', bw.seq[i], '\n')
}

###
res = FPMD(Ly = Ly, Lt = Lt, wi = NULL, Wtype = "MIX", zeta = NULL,
           bw.seq = bw.seq, NbGrid = 101, kFolds = 5, refined = TRUE,
           individualCurveJump = TRUE, nRegGrid = 51,
           npoly = 1, nder = 0, alpha = 0.05, cutoff = max, M_max = 5)

## with change points
mu_jumptime <- res$mu_jumptime
mu_jumpsize <- res$mu_jumpsize
# mu_jumpsize <- res$mu_jumpsize_h_tau
h_tau <- res$h_tau
h_d <- res$h_d
zeta <- res$zeta
wi <- res$wi
```

```
mu <- res$mu
muWork <- res$muWork
obsGrid <- res$obsGrid
workGrid <- res$workGrid


###
par(mfrow = c(1, 2))
matplot(dataSS[,100:154], type = "p", pch = 1, col = 'gray', xlab = "Day",
        ylab = "Minimum Temperature", main= "", xaxt="n")
ind_x = seq.int(1, nrow(dataSS), length.out = 6)
axis(1, at=ind_x, labels=dataSS$ymd[ind_x])
lines(obsGrid*366, mu, col = "red", lwd = 1.5)
abline( v = res$mu_jumptime*366, lty = "dashed", col = "red" )
dataSS$ymd[round(res$mu_jumptime*366)]


## combine all as iid, Xia and Qiu 2016
t = unlist(Lt);
y = unlist(Ly)[order(t)];
t = sort(t);
indJumpall = FPMD:::indMeanbreak(y = y, t = t, M_max = 15, NbGrid = 101,
                                 kernel = res$optns$kernel, npoly = 1, nder = 0)
lines(y = unique(indJumpall$mu), x = (indJumpall$obsGrid*nrow(dataSS)),
      col = "green", lty = 1, lwd = 1.5)
abline( v = round(indJumpall$mu_jumptime*nrow(dataSS)),
        lty = 'dotted', col = "green")
points(x = indJumpall$mu_jumptime*nrow(dataSS),
       y = rep(1.8, length(indJumpall$mu_jumptime)),
       pch= rep("*", length(indJumpall$mu_jumptime)),
       cex = 2, col = "green",  xpd = TRUE)
text(x = indJumpall$mu_jumptime*nrow(dataSS),
     y = rep(1.8, length(indJumpall$mu_jumptime)),
     labels = dataSS$ymd[indJumpall$mu_jumptime*nrow(dataSS)],
     xpd = TRUE, pos = 1, cex = 0.5, col = "green")


## plot mean function and confidence band
## true and estimated mean curve are based on workGrid points
plot.fmb <- function(res){

  ## confidence band
  cbandMu <- FPMD:::pwCBFun(res)
  workGrid <- res$workGrid
  muWork <- res$muWork
  rho = res$rho
  tau_est = c(0, res$mu_jumptime, 1)
  mudata = t(rbind(cbandMu, muWork))



  plot(x = NULL, y = NULL, ylim =c(7.5,21), xlim = range(Lt[[1]]),
       col = rgb(0.7,0.7,0.7,0.4), xlab = "Day",
       ylab = "Minimum Temperature", main= "", xaxt="n")
  ind_x = seq.int(1, nrow(dataSS), length.out = 6)
  axis(1, at=ind_x/366, labels=dataSS$ymd[ind_x])
  ###
```

```
  bandGrid =  t(rbind(cbandMu, workGrid))
  lband = lapply(1:(length(tau_est) - 1), function(i)
    as.data.frame(bandGrid[workGrid < (tau_est[i + 1] - rho) &
                             workGrid >= (tau_est[i] + rho),]))
 lapply(lband, function(x) polygon(c(x$workGrid, rev(x$workGrid)),c(x$lwCI,rev(x$upCI)),
                                    col = rgb(0.7,0.7,0.7,0.4) , border = NA))
  for (i in 1:(length(tau_est)-1)) {

    matplot(workGrid[workGrid < (tau_est[i+1]- rho) & workGrid >= (tau_est[i]+ rho) ],
            mudata[workGrid < (tau_est[i+1] - rho) & workGrid >= (tau_est[i]+rho), ],
            type = "l", add = TRUE, lty = c(3, 3, 1), col = c(3,3,2), lwd = 2)

  }
  legend('top', legend = c( 'Estimated mean', 'Pointwise confidence interval'),
         cex = .8, lwd = 2, col = c(2,3), lty = c(1,3), bty = "n")
  points(x = res$mu_jumptime, y = rep(7, length(res$mu_jumptime)),
         pch= rep("*", length(res$mu_jumptime)), col = 4, cex  =2, xpd = TRUE)
  text(x = res$mu_jumptime, y = rep(7, length(res$mu_jumptime)),
       labels=dataSS$ymd[round(res$mu_jumptime*366)],
       xpd = TRUE, pos = 3, cex = 0.8, col = 4)

}

plot.fmb(res)


## individual
####
year = colnames(dataSS)[-1]
par(mfrow=c(3,3))
for (i in 44:52) {

  plot(Ly[[i]],
       xlab = "Dates",
       ylab = "Average Value Weighted Returns",
       main = year[i+102],
       col = 'gray', #rgb(0.7,0.7,0.7,0.4),
       xaxt="n", ylim = c(0, max(Ly[[i]])))
  ind_x = seq.int(1, nrow(dataSS), length.out = 6)
  axis(1, at=ind_x, labels=dataSS$ymd[ind_x])
  lines(y = res$indJump[[i]]$mu, x = Lt[[i]]*nrow(dataSS),
        col = "blue", lty = 1, lwd = 1.5 )
  points(x = res$indJump[[i]]$mu_jumptime*nrow(dataSS),
         y = rep(-1, length(res$indJump[[i]]$mu_jumptime)),
         pch= rep("*", length(res$indJump[[i]]$mu_jumptime)),
         cex = 2, col = "blue",  xpd = TRUE)
  text(x = res$indJump[[i]]$mu_jumptime*nrow(dataSS),
       y = rep(-1, length(res$indJump[[i]]$mu_jumptime)),
       labels = dataSS$ymd[res$indJump[[i]]$mu_jumptime*nrow(dataSS)],
       xpd = TRUE, pos = 3, cex = 0.5, col = "blue")
  ## our
  lines(obsGrid*nrow(dataSS), mu, col = "red", lty = 1, lwd = 1.5)
  abline( v = round(res$mu_jumptime *nrow(dataSS)), col = "red", lty = "dashed")

}
par(mfrow=c(1,1))
```

```
## End(Not run)
```

---

avwreturn — *Average value weighted returns data.*

Description

The original data set consists of the daily simple returns of n = 49 industry portfolios from 1927 to 2020.

Usage

```
data(avwreturn)
```

Format

data_SS Data matrix.

Lt Observation time points.

Ly Daily average value weighted returns. ...

Source

http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

Examples

```
## Not run:

data(avwreturn)
data_SS = avwreturn$data_SS
Ly = avwreturn$Ly
Lt = avwreturn$Lt


## global test
bw.seq = seq(0.01, 0.11, by = 0.01)
reject = rep(NA, length(bw.seq))
## reject H0: no jump points
for (i in 1:length(bw.seq)) {
 reject[i] = MDTest(bw = bw.seq[i], alpha = 0.05, nRegGrid = 201, Lt = Lt, Ly = Ly, Wtype = 'MIX')
  cat('reject:', reject[i], 'for h = ', bw.seq[i], '\n')
}

###
res = FPMD(Ly = Ly, Lt = Lt, wi = NULL, Wtype = "MIX", zeta = NULL,
           bw.seq = NULL, NbGrid = 101, kFolds = 5, refined = TRUE,
```

```
            individualCurveJump = TRUE, nRegGrid = 301,
            npoly = 1, nder = 0, alpha = 0.05, cutoff = max, M_max = 15)

## with change points
mu_jumptime <- res$mu_jumptime
mu_jumpsize <- res$mu_jumpsize
# mu_jumpsize <- res$mu_jumpsize_h_tau
h_tau <- res$h_tau
h_d <- res$h_d
zeta <- res$zeta
wi <- res$wi
mu <- res$mu
muWork <- res$muWork
obsGrid <- res$obsGrid
workGrid <- res$workGrid


par(mfrow=c(1,1))
matplot(data_SS[,-1], type = "p", pch = 1, xlab = "Dates",
        ylab = "Average Value Weighted Returns",
        main = '',
        col = rgb(0.7,0.7,0.7,0.4), xaxt="n")
ind_x = seq(1, nrow(data_SS), length.out = 18)
axis(1, at=ind_x, labels=data_SS[ind_x, 1])
abline( v = round(mu_jumptime *nrow(data_SS)), lty = 2)
lines((obsGrid*nrow(data_SS)), mu, col = 4)
data_SS$Date[round(mu_jumptime*nrow(data_SS))]


## combine all as iid, Xia and Qiu 2016
t = unlist(Lt);
y = unlist(Ly)[order(t)];
t = sort(t);
indJumpall = FPMD:::indMeanbreak(y = y, t = t, M_max = 15, NbGrid = 101,
                                 kernel = res$optns$kernel, npoly = 1, nder = 0)
##  hdbinseg package from H. Cho and P. Fryzlewicz (2014) JRSSB
library(hdbinseg)
dd = t(data_SS[,-1])
ecp_CHO = dcbs.alg(dd, cp.type=1, phi=0.5, temporal=FALSE, do.parallel=0)$ecp





###
par(mfrow = c(2, 2))
ind_x = seq(1, nrow(data_SS), length.out = 18)
## 1
matplot(data_SS[1:100,-1], type = "p", pch = 1, xlab = "Dates",
        ylab = "Average Value Weighted Returns",
        main = '', col = 'gray', xaxt="n")
axis(1, at=ind_x[1:5], labels=data_SS[ind_x[1:5], 1])
abline( v = round(mu_jumptime *nrow(data_SS)), col = 'red', lty = 2)
lines((obsGrid*nrow(data_SS))[1:100], mu[1:100], col = 'red', lwd = 1.5)
abline( v = round(indJumpall$mu_jumptime*nrow(data_SS)),
        lty = 'dotted', col = "green")
lines(y = indJumpall$mu[1:100], x = (obsGrid*nrow(data_SS))[1:100],
```

```
      col = "green", lty = 1, lwd = 1.5)
points(x = round(indJumpall$mu_jumptime*nrow(data_SS))[1:2],
       y = rep(-8.3, length(indJumpall$mu_jumptime))[1:2],
       pch= rep("*", length(indJumpall$mu_jumptime)),
       cex = 2, col = "green",  xpd = TRUE)
text(x = round(indJumpall$mu_jumptime*nrow(data_SS))[1:2],
     y = rep(-8.3, length(indJumpall$mu_jumptime))[1:2],
     labels = data_SS$Date[round(indJumpall$mu_jumptime*nrow(data_SS))][1:2],
     xpd = TRUE, pos = 1, cex = 0.5, col = "green")
## cho
points(x = ecp_CHO,
       y = rep(-8.3, length(ecp_CHO)),
       pch= rep("*", length(ecp_CHO)),
       cex = 2, col = "blue",  xpd = TRUE)
text(x = ecp_CHO,
     y = rep(-8.3, length(ecp_CHO)),
     labels = data_SS$Date[ecp_CHO],
     xpd = TRUE, pos = 3, cex = 0.5, col = "blue")
## 2
matplot(data_SS[101:160,-1], type = "p", pch = 1, xlab = "Dates",
        ylab = "Average Value Weighted Returns",
        main = '', col = 'gray', xaxt="n")
axis(1, at=ind_x[6:8]-100, labels=data_SS[ind_x[6:8], 1])
abline( v = round(mu_jumptime *nrow(data_SS))[-c(1:3)]-100, col = 'red', lty = 2)
lines((obsGrid*nrow(data_SS))[1:60], mu[101:160], col = 'red', lwd = 1.5)
abline( v = round(indJumpall$mu_jumptime*nrow(data_SS))-100,
        lty = 'dotted', col = "green")
lines(y = indJumpall$mu[101:160], x = (unique(t)*nrow(data_SS))[1:60],
      col = "green", lty = 1, lwd = 1.5)
points(x = round(indJumpall$mu_jumptime*nrow(data_SS))-100,
       y = rep(-6, length(indJumpall$mu_jumptime)),
       pch= rep("*", length(indJumpall$mu_jumptime)),
       cex = 2, col = "green",  xpd = TRUE)
text(x = round(indJumpall$mu_jumptime*nrow(data_SS))-100,
     y = rep(-6, length(indJumpall$mu_jumptime)),
     labels = data_SS$Date[round(indJumpall$mu_jumptime*nrow(data_SS))],
     xpd = TRUE, pos = 1, cex = 0.5, col = "green")

## 3
matplot(data_SS[161:280,-1], type = "p", pch = 1, xlab = "Dates",
        ylab = "Average Value Weighted Returns", ylim = c(-8, 8),
        main = '', col = 'gray', xaxt="n")
axis(1, at=ind_x[9:13]-160, labels=data_SS[ind_x[9:13], 1])
abline( v = round(mu_jumptime *nrow(data_SS))[-c(1:4)]-160, col = 'red', lty = 2)
lines((obsGrid*nrow(data_SS))[1:120], mu[161:280], col = 'red', lwd = 1.5)
abline( v = round(indJumpall$mu_jumptime*nrow(data_SS))-160,
        lty = 'dotted', col = "green")
lines(y = indJumpall$mu[161:280], x = (unique(t)*nrow(data_SS))[1:120],
      col = "green", lty = 1, lwd = 1.5)
points(x = round(indJumpall$mu_jumptime*nrow(data_SS))-160,
       y = rep(-8.6, length(indJumpall$mu_jumptime)),
       pch= rep("*", length(indJumpall$mu_jumptime)),
       cex = 2, col = "green",  xpd = TRUE)
text(x = round(indJumpall$mu_jumptime*nrow(data_SS))-160,
     y = rep(-8.6, length(indJumpall$mu_jumptime)),
     labels = data_SS$Date[round(indJumpall$mu_jumptime*nrow(data_SS))],
     xpd = TRUE, pos = 1, cex = 0.5, col = "green")
```

```
## 4
matplot(data_SS[281:354,-1], type = "p", pch = 1, xlab = "Dates",
        ylab = "Average Value Weighted Returns",
        main = '', col = 'gray', xaxt="n")
axis(1, at=ind_x[14:18]-280, labels=data_SS[ind_x[14:18], 1])
abline( v = round(mu_jumptime *nrow(data_SS))[-c(1:7)]-280, col = 'red', lty = 2)
lines((obsGrid*nrow(data_SS))[1:74], mu[281:354], col = 'red', lwd = 1.5)
lines(y = indJumpall$mu[281:354], x = (unique(t)*nrow(data_SS))[1:74],
      col = "green", lty = 1, lwd = 1.5)
abline( v = round(indJumpall$mu_jumptime*nrow(data_SS))-280,
        lty = 'dotted', col = "green")
points(x = indJumpall$mu_jumptime*nrow(data_SS) - 280,
       y = rep(-24.4, length(indJumpall$mu_jumptime)),
       pch= rep("*", length(indJumpall$mu_jumptime)),
       cex = 2, col = "green",  xpd = TRUE)
text(x = indJumpall$mu_jumptime*nrow(data_SS) - 280,
     y = rep(-24.4, length(indJumpall$mu_jumptime)),
     labels = data_SS$Date[indJumpall$mu_jumptime*nrow(data_SS)],
     xpd = TRUE, pos = 1, cex = 0.5, col = "green")
## cho
points(x = ecp_CHO - 280,
       y = rep(-24.4, length(ecp_CHO)),
       pch= rep("*", length(ecp_CHO)),
       cex = 2, col = "blue",  xpd = TRUE)
text(x = ecp_CHO - 280,
     y = rep(-24.4, length(ecp_CHO)),
     labels = data_SS$Date[ecp_CHO],
     xpd = TRUE, pos = 3, cex = 0.5, col = "blue")



## plot mean function and confidence band
## true and estimated mean curve are based on workGrid points
plot.fmb <- function(res){

  ## confidence band
  cbandMu <- FPMD:::pwCBFun(res)
  workGrid <- res$workGrid
  muWork <- res$muWork
  rho = res$rho
  tau_est = c(0, res$mu_jumptime, 1)
  mudata = t(rbind(cbandMu, muWork))

  ###
  bandGrid =  t(rbind(cbandMu, workGrid))
  lband = lapply(1:(length(tau_est) - 1), function(i)
    as.data.frame(bandGrid[workGrid < (tau_est[i + 1] - rho) &
                             workGrid >= (tau_est[i] + rho),]))

  par(mfrow = c(2, 2))
  ind_x = seq(1, nrow(data_SS), length.out = 18)
  ## 1
  plot(x = NULL, y = NULL, ylim = c(-3, 3), xlim = c(0, 0.26),
       xlab = "Dates",
       ylab = "Average Value Weighted Returns",
       main = '',
```

```
     col = rgb(0.7,0.7,0.7,0.4), xaxt="n")
axis(1, at= (ind_x[1:5]/nrow(data_SS)), labels=data_SS[ind_x[1:5], 1])

lapply(lband[1:5], function(x) polygon(c(x$workGrid, rev(x$workGrid)),c(x$lwCI,rev(x$upCI)),
                                        col = rgb(0.7,0.7,0.7,0.4) , border = NA))
for (i in 1:5) {
  matplot(workGrid[workGrid < (tau_est[i+1]- rho) & workGrid >= (tau_est[i]+ rho) ],
          mudata[workGrid < (tau_est[i+1] - rho) & workGrid >= (tau_est[i]+rho), ],
          type = "l", add = TRUE, lty = c(3, 3, 1), col = c(4,4,2), lwd = 2)

}
legend('top', legend = c( 'Estimated mean', 'Pointwise confidence interval'),
       cex = 1, lwd = 2, col = c(2,4), lty = c(1,3), bty = "n")
points(x = res$mu_jumptime[1:4], y = rep(-3.2, length(res$mu_jumptime[1:4])),
       pch= rep("*", length(res$mu_jumptime[1:4])), cex = 2, col = 4,  xpd = TRUE)
# lines(res$obsGrid, res$mu, col = 4)
text(x = res$mu_jumptime[1:4], y = rep(-3.2, length(res$mu_jumptime[1:4])),
     labels = data_SS$Date[round(res$mu_jumptime[1:4]*nrow(data_SS))],
     xpd = TRUE, pos = 3, cex = 0.8, col = 4)

## 2
plot(x = NULL, y = NULL, ylim = c(-1.5, 1.5), xlim = c(0.26, 0.46),
     xlab = "Dates",
     ylab = "Average Value Weighted Returns",
     main = '',
     col = rgb(0.7,0.7,0.7,0.4), xaxt="n")
axis(1, at= (ind_x[5:8]/nrow(data_SS)), labels=data_SS[ind_x[5:8], 1])
###
lapply(lband[5:8], function(x) polygon(c(x$workGrid, rev(x$workGrid)),c(x$lwCI,rev(x$upCI)),
                                        col = rgb(0.7,0.7,0.7,0.4) , border = NA))
for (i in 5:8) {
  matplot(workGrid[workGrid < (tau_est[i+1]- rho) & workGrid >= (tau_est[i]+ rho) ],
          mudata[workGrid < (tau_est[i+1] - rho) & workGrid >= (tau_est[i]+rho), ],
          type = "l", add = TRUE, lty = c(3, 3, 1), col = c(4,4,2), lwd = 2)
}
legend('top', legend = c( 'Estimated mean', 'Pointwise confidence interval'),
       cex = 1, lwd = 2, col = c(2,4), lty = c(1,3), bty = "n")
points(x = res$mu_jumptime[5:7], y = rep(-1.6, length(res$mu_jumptime[5:7])),
       pch= rep("*", length(res$mu_jumptime[5:7])), cex = 2, col = 4,  xpd = TRUE)
# lines(res$obsGrid, res$mu, col = 4)
text(x = res$mu_jumptime[5:7], y = rep(-1.6, length(res$mu_jumptime[5:7])),
     labels = data_SS$Date[round(res$mu_jumptime[5:7]*nrow(data_SS))],
     xpd = TRUE, pos = 3, cex = 0.8, col = 4)


## 3
plot(x = NULL, y = NULL, ylim = c(-1.5, 1.5), xlim = c(0.46, 0.78),
     xlab = "Dates",
     ylab = "Average Value Weighted Returns",
     main = '',
     col = rgb(0.7,0.7,0.7,0.4), xaxt="n")
axis(1, at= (ind_x[8:13]/nrow(data_SS)), labels=data_SS[ind_x[8:13], 1])
###
###
lapply(lband[8:11], function(x) polygon(c(x$workGrid, rev(x$workGrid)),c(x$lwCI,rev(x$upCI)),
                                         col = rgb(0.7,0.7,0.7,0.4) , border = NA))
for (i in 8:11) {
```

```
    matplot(workGrid[workGrid < (tau_est[i+1]- rho) & workGrid >= (tau_est[i]+ rho) ],
            mudata[workGrid < (tau_est[i+1] - rho) & workGrid >= (tau_est[i]+rho), ],
            type = "l", add = TRUE, lty = c(3, 3, 1), col = c(4,4,2), lwd = 2)
  }
  legend('top', legend = c( 'Estimated mean', 'Pointwise confidence interval'),
         cex = 1, lwd = 2, col = c(2,4), lty = c(1,3), bty = "n")
  points(x = res$mu_jumptime[8:10], y = rep(-1.6, length(res$mu_jumptime[8:10])),
         pch= rep("*", length(res$mu_jumptime[8:10])), cex = 2, col = 4,  xpd = TRUE)
  # lines(res$obsGrid, res$mu, col = 4)
  text(x = res$mu_jumptime[8:10], y = rep(-1.6, length(res$mu_jumptime[8:10])),
       labels = data_SS$Date[round(res$mu_jumptime[8:10]*nrow(data_SS))],
       xpd = TRUE, pos = 3, cex = 0.7, col = 4)

  ## 4
  plot(x = NULL, y = NULL, ylim = c(-4, 5), xlim = c(0.78, 1),
       xlab = "Dates",
       ylab = "Average Value Weighted Returns",
       main = '',
       col = rgb(0.7,0.7,0.7,0.4), xaxt="n")
  axis(1, at= (ind_x[12:18]/nrow(data_SS)), labels=data_SS[ind_x[12:18], 1])
 lapply(lband[11:14], function(x) polygon(c(x$workGrid, rev(x$workGrid)),c(x$lwCI,rev(x$upCI)),
                                          col = rgb(0.7,0.7,0.7,0.4) , border = NA))
  for (i in 11:14) {
    matplot(workGrid[workGrid < (tau_est[i+1]- rho) & workGrid >= (tau_est[i]+ rho) ],
            mudata[workGrid < (tau_est[i+1] - rho) & workGrid >= (tau_est[i]+rho), ],
            type = "l", add = TRUE, lty = c(3, 3, 1), col = c(4,4,2), lwd = 2)
  }
  legend('topleft', legend = c( 'Estimated mean', 'Pointwise confidence interval'),
         cex = 1, lwd = 2, col = c(2,4), lty = c(1,3), bty = "n")
  points(x = res$mu_jumptime[11:13], y = rep(-4.3, length(res$mu_jumptime[11:13])),
         pch= rep("*", length(res$mu_jumptime[11:13])), cex = 2, col = 4,  xpd = TRUE)
  # lines(res$obsGrid, res$mu, col = 4)
  text(x = res$mu_jumptime[11:13], y = rep(-4.3, length(res$mu_jumptime[11:13])),
       labels = data_SS$Date[round(res$mu_jumptime[11:13]*nrow(data_SS))],
       xpd = TRUE, pos = 3, cex = 0.8, col = 4)


}

plot.fmb(res)


## individual, Xia and Qiu 2016
#### for each curve 1,5,6,16,35, 36
industry = colnames(data_SS)[-50]
par(mfrow=c(2,2))
for (i in c(5,16,35, 36)) {
  plot(data_SS[, i+1],
       xlab = "Dates",
       ylab = "Average Value Weighted Returns",
       main = industry[i+1],
       col = 'gray', #rgb(0.7,0.7,0.7,0.4),
       xaxt="n",
       ylim = c(-6,6))
  ind_x = seq(1, nrow(data_SS), length.out = 10)
  axis(1, at= ind_x, labels = data_SS[ind_x, 1])
  ## indivudual
```

```
  lines(y = res$indJump[[i]]$mu, x = Lt[[i]]*nrow(data_SS),
        col = "blue", lty = 1, lwd = 1.5)
  points(x = res$indJump[[i]]$mu_jumptime*nrow(data_SS),
         y = rep(-6.4, length(res$indJump[[i]]$mu_jumptime)),
         pch= rep("*", length(res$indJump[[i]]$mu_jumptime)),
         cex = 2, col = "blue",  xpd = TRUE)
  text(x = res$indJump[[i]]$mu_jumptime*nrow(data_SS),
       y = rep(-6.4, length(res$indJump[[i]]$mu_jumptime)),
       labels = data_SS$Date[res$indJump[[i]]$mu_jumptime*nrow(data_SS)],
       xpd = TRUE, pos = 3, cex = 0.5, col = "blue")
  ## our mean
  lines(obsGrid*nrow(data_SS), mu, col = "red", lty = 1, lwd = 1.5)
  abline( v = round(mu_jumptime *nrow(data_SS)), col = "red", lty = "dashed")


}
par(mfrow=c(1,1))

## End(Not run)
```

FPMD *Functional Processes with Multiple Discontinuities*

Description

Multiple Thresholds detection in mean function for dense or sparse functional data. The bandwidths are selected by a k-folds cross validation.

Usage

```
FPMD(
  Ly,
  Lt,
  wi = NULL,
  Wtype = c("OBS", "SUBJ", "MIX", "OPT"),
  zeta = NULL,
  cutoff = max,
  alpha = 0.05,
  bw.seq = NULL,
  nRegGrid = 101,
  NbGrid = 101,
  kernel = "epan",
  npoly = 1,
  nder = 0,
  kFolds = 5,
  refined = FALSE,
  individualCurveJump = FALSE,
  M_max
)
```

Arguments

Ly	A list of n vectors containing the observed values for each individual. Missing values specified by NAs are supported for dense case (dataType='dense').
Lt	A list of n vectors containing the observation time points for each individual corresponding to y. Each vector should be sorted in ascending order.
wi	A vector of weight for each observation.
Wtype	Weight structure specified for estimation. It works if wi = NULL.
zeta	Cut-off threshold for change point detection. The default is NULL.
cutoff	The method for construct the cut-off statistics. Default is max. For the case observations are much larger than subjections, we suggest to use the min.
alpha	The confidence level for the cut-off threshold. The default is alpha = 0.05, and for heavy tailed process, we suggest a higher level 0.1.
bw.seq	The selected (or user specified) bandwidth for cross validation. The default is NULL.
nRegGrid	The number of support points in each direction of covariance surface; numeric - default: 101.
NbGrid	The number of support points in the jump points detection; numeric - default: 101.
kernel	Smoothing kernel choice, common for mu and covariance; "rect", "epan", "quar" - default: "epan".
npoly	The degree of polynomial. Default is 1 for local linear smoothing.
nder	The order of derivative. Default is 0 for local linear smoothing, and should smaller than npoly.
kFolds	Number of folds for cross validation - default: 5.
refined	A refined stage for jump size estimation after the jump locations are detected; logical - default: TRUE
individualCurveJump	Jump detection for each curve by the BIC proposed in Xia and Qiu, 2016.
M_max	The maximum number of jumps for individual curve when using the BIC.

Value

A list containing the following fields: If the change points are detected, the function output the followings

mu_jumptime	The detected change point locations. If no change point detected, mu_jumptime=0
mu_jumpsize	The estimated jump sizes. If no change point detected, mu_jumpsize=0
mu_jumpsize_h_tau	The detected change point locations. If no change point detected, mu_jumptime=0
zeta	The cut-off threshold for change point detection.
mu	For no change point case. A vector of length obsGrid containing the mean function estimate.
muWork	For change point case. A vector of length obsGrid containing the mean function estimate.
wi	The weight for each observation.
mi	Number of observations for each subject.

`timings`	A vector with execution times for the basic parts of the FMbreaks call.
`timings_jump`	A vector with execution times for the basic parts of jump detection.
`workGrid`	A vector of length nWorkGrid.
`regGrid`	A vector of length regGrid.
`sigma2`	Variance for measure error.
`smoothCov`	A nWorkGrid by nWorkGrid matrix of the smoothed covariance surface.
`fittedCov`	A nWorkGrid by nWorkGrid matrix of the fitted covariance surface, which is guaranteed to be non-negative definite.
`optns`	A list of actually used options.
`h_tau`	The selected (or user specified) bandwidth for jump locations estimate.
`h_d`	The selected (or user specified) bandwidth for jump sizes estiamte.
`LX`	The fitted individual curves.
`indJump`	The jump detection for each individual curve.

References

Li, J., Li, Y., and Hsing, T. (2021). " On Functional Processes with Multiple Discontinuities".

Examples

```
## Not run:
# setting 1
## 3 change points
tau = c(.25,.5,.75) # jump locations
disc = c(.5,-.4,.4) # jump sizes
smoothmean=function(x) x^2+sin(2*pi*x) + cos(2*pi*x)
jump = function(x) sum((x>=tau)*disc) # jump function
mu_fun= function(t){smoothmean(t)+ sapply(t, jump)}
n = 400
Q = 20

# set.seed(123)
## generate gaussian process
## "cos", "sin", "fourier", "legendre01", "poly"
data = MakeFPMBData(n = n, Q = Q, proc= "gaussian", muFun = mu_fun, K = 3,
                    lambda = 1/(seq_len(3)+1)^2, sigma = 0.2, basisType = "fourier")
Lt = data$Lt
Ly = data$Ly
bw.seq = seq(0.01, 0.13, by = 0.02)
system.time(resCP <- FPMD(Ly = Ly, Lt = Lt, wi = NULL, Wtype = "MIX", zeta = NULL,
                              bw.seq = 0.06, NbGrid = 101, kFolds = 5, refined = TRUE,
                              individualCurveJump = FALSE, nRegGrid = 101,
                              npoly = 1, nder = 0, alpha = 0.05, cutoff = max))
h_tau = resCP$h_tau
h_d = resCP$h_d
zeta = resCP$zeta
mu_jumptime = resCP$mu_jumptime
mu_jumpsize = resCP$mu_jumpsize

## pointwise confidence band
PCBplot(res = resCP)
lines(resCP$obsGrid, mu_fun(resCP$obsGrid), type="l", col = 'blue' )
```

```
## End(Not run)
```

MakeFPMBData *Create a Dense Functional Data sample for a Gaussian process*

Description

For a Gaussian or t process, create a dense or sparse functional data sample of size n over a [0,1] support.

Usage

```
MakeFPMBData(
  n,
  Q,
  muFun,
  rdist = runif,
  K,
  lambda,
  sigma,
  basisType = "cos",
  proc = c("gaussian", "t")
)
```

Arguments

`n`	number of samples to generate.
`Q`	an integral. The observations for per sample are following the poisson distribution with mean Q.
`muFun`	a function that takes a vector input and output a vector of the corresponding mean (default: zero function).
`rdist`	a sampler for generating the random design time points within [0, 1].
`K`	scalar specifying the number of basis to be used (default: 3).
`lambda`	vector of size K specifying the variance of each components (default: rep(1,K)).
`sigma`	The standard deviation of the Gaussian noise added to each observation points.
`basisType`	string specifying the basis type used; possible options are: 'sin', 'cos' and 'fourier' (default: 'cos') (See code of 'CreateBasis' for implementation details.)
`proc`	The type of random process, 'gaussian' or 't' process.

Value

A list containing the generated data with `List` type:

`Ly`	A list of *n* vectors containing the observed values for each individual. Missing values specified by `NA`s are supported for dense case (`dataType='dense'`).

`Lt`	A list of n vectors containing the observation time points for each individual corresponding to y. Each vector should be sorted in ascending order.
`mi`	Number of observations for each subject.

`Maturities` *Monthly U.S. Treasuries Data.*

Description

The monthly U.S. treasuries from January 1983 to September 2010. In total, there are n = 15 interest rates at maturities of 3, 6, 9, 12, 18, 24, 30, 36, 48, 60, 72, 84, 96, 108, and 120 months.

Usage

```
data(Maturities)
```

Format

dmat Data matrix.

Lt Observation time points.

Ly Monthly maturity. ...

Examples

```
## Not run:
data(Maturities)
dmat = Maturities$dmat
Ly = Maturities$Ly
Lt = Maturities$Lt


## global test
bw.seq = seq(0.01, 0.11, by = 0.02)
reject = rep(NA, length(bw.seq))
## reject H0: no jump points
for (i in 1:length(bw.seq)) {
  reject[i] = MDTest(bw = bw.seq[i], alpha = 0.05, Lt = Lt, Ly = Ly, Wtype = 'MIX')
  cat('reject:', reject[i], 'for h = ', bw.seq[i], '\n')
}

##
res = FPMD(Ly = Ly, Lt = Lt, wi = NULL, Wtype = "MIX", zeta = NULL,
               bw.seq = NULL, NbGrid = 101, kFolds = 5, refined = TRUE,
               individualCurveJump = TRUE, nRegGrid = 101,
               npoly = 1, nder = 0, alpha = 0.05, cutoff = max, M_max = 10)


## with change points
mu_jumptime <- res$mu_jumptime
mu_jumpsize <- res$mu_jumpsize
# mu_jumpsize <- res$mu_jumpsize_h_tau
h_tau <- res$h_tau
h_d <- res$h_d
```

```
zeta <- res$zeta
wi <- res$wi
mu <- res$mu
muWork <- res$muWork
obsGrid <- res$obsGrid
workGrid <- res$workGrid


library(zoo)
ym <- seq(as.yearmon("1983-01"), as.yearmon("2010-09"), 1/12)
ym = format(as.Date(ym), "%Y-%m")


par(mfrow=c(1,2))
matplot(dmat, type = "p", pch = 1,
        xlab = "Dates", ylab = "Maturities(Months)",
        #main = 'Monthly U.S. Treasuries',
        col = 'gray', xaxt="n")
ind_x = c(1, round(mu_jumptime*333), 333)
axis(1, at=ind_x, labels= ym[ind_x])
abline( v = round(mu_jumptime*333), lty = "dashed", col = "red")
lines(obsGrid*333, mu, col = "red", lwd = 1.5)


## combine all as iid, Xia and Qiu 2016
t = unlist(Lt);
y = unlist(Ly)[order(t)];
t = sort(t);
indJumpall = FPMD:::indMeanbreak(y = y, t = t, M_max = 15, NbGrid = 101,
                                 kernel = res$optns$kernel, npoly = 1, nder = 0)
lines(y = indJumpall$mu, x = (unique(t)*nrow(dmat)),
      col = "green", lty = 1, lwd = 1.5)
abline(v = round(indJumpall$mu_jumptime*nrow(dmat)),
       lty = 'dotted', col = "green")
points(x = indJumpall$mu_jumptime*nrow(dmat),
       y = rep(-.55, length(indJumpall$mu_jumptime)),
       pch= rep("*", length(indJumpall$mu_jumptime)),
       cex = 2, col = "green",  xpd = TRUE)
text(x = indJumpall$mu_jumptime*nrow(dmat),
     y = rep(-.55, length(indJumpall$mu_jumptime)),
     labels = ym[indJumpall$mu_jumptime*nrow(dmat)],
     xpd = TRUE, pos = 1, cex = 0.5, col = "green")
## hdbinseg package from H. Cho and P. Fryzlewicz (2014) JRSSB
library(hdbinseg)
dd = t(dmat)
ecp_CHO = dcbs.alg(dd, cp.type=1, phi= 1, temporal = TRUE, do.parallel=0)$ecp
# ecp_CHO = sbs.alg(dd, cp.type=1, temporal = TRUE, do.parallel=0)$ecp
## cho
points(x = ecp_CHO,
       y = rep(-0.55, length(ecp_CHO)),
       pch= rep("*", length(ecp_CHO)),
       cex = 2, col = "blue",  xpd = TRUE)
text(x = ecp_CHO,
     y = rep(-0.55, length(ecp_CHO)),
     labels = ym[ecp_CHO],
     xpd = TRUE, pos = 3, cex = 0.5, col = "blue")
```

```
## plot mean function and confidence band
## true and estimated mean curve are based on workGrid points
plot.fmb <- function(res){

  ## confidence band
  cbandMu <- FPMD:::pwCBFun(res)
  workGrid <- res$workGrid
  muWork <- res$muWork
  rho = res$rho
  tau_est = c(0, res$mu_jumptime, 1)
  mudata = t(rbind(cbandMu, muWork))


  plot(x = NULL, y = NULL, ylim = c(0, 11), xlim = range(unlist(Lt)),
          xlab = "Dates",
          ylab = "Maturities(Months)",
          main = '',
          col = rgb(0.7,0.7,0.7,0.4), xaxt="n")
  ind_x = c(min(Lt[[1]]), res$mu_jumptime, max(Lt[[1]]))
  axis(1, at=ind_x, labels= ym[round(ind_x*333)])
  ###

  bandGrid =  t(rbind(cbandMu, workGrid))
  lband = lapply(1:(length(tau_est) - 1), function(i)
    as.data.frame(bandGrid[workGrid < (tau_est[i + 1] - rho) &
                             workGrid >= (tau_est[i] + rho),]))
 lapply(lband, function(x) polygon(c(x$workGrid, rev(x$workGrid)),c(x$lwCI,rev(x$upCI)),
                                    col = rgb(0.7,0.7,0.7,0.4) , border = NA))
  for (i in 1:(length(tau_est)-1)) {

    matplot(workGrid[workGrid < (tau_est[i+1]- rho) & workGrid >= (tau_est[i]+ rho) ],
            mudata[workGrid < (tau_est[i+1] - rho) & workGrid >= (tau_est[i]+rho), ],
            type = "l", add = TRUE, lty = c(3, 3, 1), col = c(3,3,2), lwd = 2)

  }
  legend('top', legend = c( 'Estimated mean', 'Pointwise confidence interval'),
         lwd = 2, col = c(2,3), lty = c(1,3), bty = "n")
  points(x = res$mu_jumptime, y = rep(-.4, length(res$mu_jumptime)),
         pch= rep("*", length(res$mu_jumptime)), col = 4, cex  =2, xpd = TRUE)
  # # lines(res$obsGrid, res$mu, col = 2)
  text(x = res$mu_jumptime, y = rep(-.4, length(res$mu_jumptime)),
       labels= ym[round(res$mu_jumptime*nrow(dmat))],
       xpd = TRUE, pos = 3, cex = 0.8, col = 4)

}

plot.fmb(res)

## individual
####
par(mfrow=c(3,5))
ind = c(3, 6, 9, 12, 18, 24, 30, 36, 48, 60, 72, 84, 96, 108, 120)
for (i in 1:length(ind)) {

  plot(dmat[, i],
```

```
      xlab = "Dates",
      ylab = "Maturities",
      main = paste("maturities of", ind[i], "months"),
      col = 'gray', #rgb(0.7,0.7,0.7,0.4),
      xaxt="n")
  ind_x = c(1, round(mu_jumptime*333), 333)
  axis(1, at=ind_x, labels= ym[ind_x])
  lines(y = res$indJump[[i]]$mu, x = Lt[[i]]*333,
        col = "blue", lty = 1, lwd = 1.5 )
  points(x = round(res$indJump[[i]]$mu_jumptime*333),
         y = rep(min(dmat[, i])-0.45, length(res$indJump[[i]]$mu_jumptime)),
         pch= rep("*", length(res$indJump[[i]]$mu_jumptime)),
         cex = 2, col = "blue", xpd = TRUE)
  text(x = round(res$indJump[[i]]$mu_jumptime*333),
      y = rep(min(dmat[, i])-0.45, length(res$mu_jumptime)),
      labels= ym[res$indJump[[i]]$mu_jumptime*nrow(dmat)],
      xpd = TRUE, pos = 3, cex = 0.5, col = "blue")
  ## our
  lines(obsGrid*333, mu, col = "red", lty = 1, lwd = 1.5)
  abline( v = round(mu_jumptime *nrow(dmat)), col = "red", lty = "dashed")

}
par(mfrow=c(1,1))




## End(Not run)
```

MDTest *A global test for jump points in mean function*

Description

A global test for jump points in mean function

Usage

```
MDTest(
  bw,
  alpha = 0.05,
  Lt,
  Ly,
  nRegGrid = 101,
  Wtype = c("OBS", "SUBJ", "MIX", "OPT")
)
```

Arguments

bw	The selected (or user specified) bandwidth.
alpha	The confidence level for the cut-off threshold. The default is `alpha = 0.05`, and for heavy tailed process, we suggest a higher level `0.1`.

Lt	A list of n vectors containing the observation time points for each individual corresponding to y. Each vector should be sorted in ascending order.
Ly	A list of n vectors containing the observed values for each individual. Missing values specified by NAs are supported for dense case (dataType='dense').
nRegGrid	The number of support points in each direction of covariance surface; numeric - default: 101.
Wtype	Weight structure specified for estimation. It works if wi = NULL.

Value

reject	1 indicates that there exists discontinuities.

MeanBreaksFP *Multiple Thresholds detection in mean function*

Description

Multiple Thresholds detection in mean function for dense or sparse functional data.

Usage

```
MeanBreaksFP(
  yin,
  xin,
  win,
  mi,
  xout,
  NbGrid,
  h_tau,
  h_d,
  zeta,
  npoly = 1,
  nder = 0,
  kernel,
  refined = FALSE
)
```

Arguments

yin	A vector containing the observed values.
xin	A vector containing the observation time points corresponding to yin.
win	A list containing a vector of weight for each observations and the duplicated weight.
mi	A vector containing the number of observations for each subject.
xout	A vector containing the observation time points for mean estimation.
NbGrid	The number of support points in the jump points detection; numeric - default: 101.
h_tau	The selected (or user specified) bandwidth for jump locations estimate.

h_d	The selected (or user specified) bandwidth for jump sizes estiamte.
zeta	Cut-off threshold for change point detection.
npoly	The degree of polynomial. Default is 1 for local linear smoothing.
nder	The order of derivative. Default is 0 for local linear smoothing, and should smaller than npoly.
kernel	Smoothing kernel choice, common for mu and covariance; "rect", "epan", "quar" - default: "epan".
refined	A refined stage for jump size estimation after the jump locations are detected; logical - default: TRUE

Value

A list containing the following fields: If the change points are detected, the function output the followings

mu_jumptime	The detected change point locations. If no change point detected, `mu_jumptime=0`
mu_jumpsize	The estimated jump sizes. If no change point detected, `mu_jumpsize=0`
mu_jumpsize_h_tau	The detected change point locations. If no change point detected, `mu_jumptime=0`
mu	For no change point case. A vector of length obsGrid containing the mean function estimate.
muout	For change point case. A vector of length xout containing the mean function estimate.
xout	A vector containing the observation time points for mean estimation.
timings	A vector with execution times for the basic parts of the FMbreaks call.
h_tau	The selected (or user specified) bandwidth for jump locations estimate.
h_d	The selected (or user specified) bandwidth for jump sizes estiamte.

References

Li, J., Li, Y., and Hsing, T. (2021). " On Functional Processes with Multiple Discontinuities".

MMavwreturn *Monthly max average value weighted returns data.*

Description

The original data set consists of the daily simple returns of n = 49 industry portfolios from 1927 to 2020.

Usage

```
data(MMavwreturn)
```

Format

data_SS Data matrix.

Lt Observation time points.

Ly Daily average value weighted returns. ...

Source

http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

Examples

```
## Not run:



library(reshape2)
library(zoo)
library(dplyr)

data(MMavwreturn)
data_SS = MMavwreturn$data_SS
Ly = MMavwreturn$Ly
Lt = MMavwreturn$Lt

## global test
bw.seq = seq(0.01, 0.11, by = 0.02)
reject = rep(NA, length(bw.seq))
## reject H0: no jump points
for (i in 1:length(bw.seq)) {
  reject[i] = MDTest(bw = bw.seq[i], alpha = 0.05, Lt = Lt, Ly = Ly, Wtype = 'MIX')
  cat('reject:', reject[i], 'for h = ', bw.seq[i], '\n')
}


##
res = FPMD(Ly = Ly, Lt = Lt, wi = NULL, Wtype = "MIX", zeta = NULL,
           bw.seq = NULL, NbGrid = 101, kFolds = 5, refined = TRUE,
           individualCurveJump = TRUE, nRegGrid = 501,
           npoly = 1, nder = 0, alpha = 0.05, cutoff = max, M_max = 15)

## with change points
mu_jumptime <- res$mu_jumptime
mu_jumpsize <- res$mu_jumpsize
# mu_jumpsize <- res$mu_jumpsize_h_tau
h_tau <- res$h_tau
h_d <- res$h_d
zeta <- res$zeta
wi <- res$wi
mu <- res$mu
muWork <- res$muWork
obsGrid <- res$obsGrid
workGrid <- res$workGrid


par(mfrow = c(1, 2))
matplot(data_SS[,-50], type = "p", pch = 1, xlab = "Dates",
        ylab = "Average Value Weighted Returns",
        main = '',
        col = 'gray', xaxt="n")
ind_x = seq(1, nrow(data_SS), length.out = 10)
axis(1, at=ind_x, labels=data_SS[ind_x, 50])
abline( v = round(mu_jumptime *nrow(data_SS)), lty = "dashed", col = "red")
```

```
lines(obsGrid*nrow(data_SS), mu, col = "red", lwd = 1.5)
data_SS$ym[round(mu_jumptime *nrow(data_SS))]



## combine all as iid, Xia and Qiu 2016
t = unlist(Lt);
y = unlist(Ly)[order(t)];
t = sort(t);
indJumpall = FPMD:::indMeanbreak(y = y, t = t, M_max = 15, NbGrid = 101,
                          kernel = res$optns$kernel, npoly = 1, nder = 0)
lines(y = indJumpall$mu, x = (unique(t)*nrow(data_SS)),
      col = "green", lty = 1, lwd = 1.5)
abline( v = round(indJumpall$mu_jumptime*nrow(data_SS)),
        lty = 'dotted', col = "green")
points(x = round(indJumpall$mu_jumptime*nrow(data_SS)),
       y = rep(-.7, length(indJumpall$mu_jumptime)),
       pch= rep("*", length(indJumpall$mu_jumptime)),
       cex = 2, col = "green",  xpd = TRUE)
text(x = round(indJumpall$mu_jumptime*nrow(data_SS)),
     y = rep(-.7, length(indJumpall$mu_jumptime)),
     labels = data_SS$ym[round(indJumpall$mu_jumptime*nrow(data_SS))],
     xpd = TRUE, pos = 1, cex = 0.5, col = "green")
## hdbinseg package from H. Cho and P. Fryzlewicz (2014) JRSSB
library(hdbinseg)
dd = t(data_SS[,-50])
ecp_CHO = dcbs.alg(dd, cp.type=1, phi= -1, temporal=TRUE, do.parallel=0)$ecp
# ecp_CHO = sbs.alg(dd, cp.type=1, temporal = TRUE, do.parallel=0)$ecp
## cho
points(x = ecp_CHO,
       y = rep(-0.7, length(ecp_CHO)),
       pch= rep("*", length(ecp_CHO)),
       cex = 2, col = "blue",  xpd = TRUE)
text(x = ecp_CHO,
     y = rep(-0.7, length(ecp_CHO)),
     labels = data_SS$ym[ecp_CHO],
     xpd = TRUE, pos = 3, cex = 0.5, col = "blue")



## plot mean function and confidence band
## true and estimated mean curve are based on workGrid points
plot.fmb <- function(res){

  ## confidence band
  cbandMu <- FPMD:::pwCBFun(res)
  workGrid <- res$workGrid
  muWork <- res$muWork
  rho = res$rho
  tau_est = c(0, res$mu_jumptime, 1)
  mudata = t(rbind(cbandMu, muWork))


  plot(x = NULL, y = NULL, ylim = c(0, 9), xlim = range(unlist(Lt)),
       xlab = "Dates",
       ylab = "Average Value Weighted Returns",
       main = '',
```

```
        col = rgb(0.7,0.7,0.7,0.4), xaxt="n")
  ind_x = seq(min(Lt[[1]]), max(Lt[[1]]), length.out = 10)
  axis(1, at=ind_x, labels= data_SS[ind_x*nrow(data_SS), 50])
  ###

  bandGrid =  t(rbind(cbandMu, workGrid))
  lband = lapply(1:(length(tau_est) - 1), function(i)
    as.data.frame(bandGrid[workGrid < (tau_est[i + 1] - rho) &
                             workGrid >= (tau_est[i] + rho),]))
 lapply(lband, function(x) polygon(c(x$workGrid, rev(x$workGrid)),c(x$lwCI,rev(x$upCI)),
                                    col = rgb(0.7,0.7,0.7,0.4) , border = NA))
  for (i in 1:(length(tau_est)-1)) {

    matplot(workGrid[workGrid < (tau_est[i+1]- rho) & workGrid >= (tau_est[i]+ rho) ],
            mudata[workGrid < (tau_est[i+1] - rho) & workGrid >= (tau_est[i]+rho), ],
            type = "l", add = TRUE, lty = c(3, 3, 1), col = c(4,4,2), lwd = 2)

  }
  legend('topleft', legend = c( 'Estimated mean', 'Pointwise confidence interval'),
         lwd = 2, col = c(2,4), lty = c(1,3), bty = "n")
  points(x = res$mu_jumptime, y = rep(-.35, length(res$mu_jumptime)),
         pch= rep("*", length(res$mu_jumptime)), cex = 2, col = 4,  xpd = TRUE)
  # lines(res$obsGrid, res$mu, col = 4)
  text(x = res$mu_jumptime, y = rep(-.3, length(res$mu_jumptime)),
       labels = data_SS$ym[round(res$mu_jumptime*nrow(data_SS))],
       xpd = TRUE, pos = 3, cex = 0.5, col = 4)

}

plot.fmb(res)

## individual
####
industry = colnames(data_SS)[-50]
par(mfrow=c(2,2))
for (i in c(5,16,35, 36)) {
  plot(data_SS[, i],
       xlab = "Dates",
       ylab = "Average Value Weighted Returns",
       main = industry[i],
       col = 'gray', #rgb(0.7,0.7,0.7,0.4),
       xaxt="n", ylim = c(0, max(data_SS[, i])))
  ind_x = seq(1, nrow(data_SS), length.out = 10)
  axis(1, at=ind_x, labels=data_SS[ind_x, 50])
  ## indivudual
  lines(y = res$indJump[[i]]$mu, x = Lt[[i]]*nrow(data_SS),
        col = "blue", lty = 1, lwd = 1.5 )
  points(x = res$indJump[[i]]$mu_jumptime*nrow(data_SS),
         y = rep(-0.55, length(res$indJump[[i]]$mu_jumptime)),
         pch= rep("*", length(res$indJump[[i]]$mu_jumptime)),
         cex = 2, col = "blue",  xpd = TRUE)
  text(x = res$indJump[[i]]$mu_jumptime*nrow(data_SS),
       y = rep(-0.55, length(res$indJump[[i]]$mu_jumptime)),
       labels = data_SS$ym[res$indJump[[i]]$mu_jumptime*nrow(data_SS)],
       xpd = TRUE, pos = 3, cex = 0.5, col = "blue")
  ## our
  lines(obsGrid*nrow(data_SS), mu, col = "red", lty = 1, lwd = 1.5)
```

```
  abline( v = round(mu_jumptime *nrow(data_SS)), col = "red", lty = "dashed")
}


## End(Not run)
```

`PCBplot` *Plot the estimated mean function and the corresponding pointwise confidence interval*

Description

Plot the estimated mean function and the corresponding pointwise confidence interval

Usage

```
PCBplot(res)
```

Arguments

`res`	an FPMD class object returned by `FPMD()`.

Index