Package 'fTB'

May 13, 2016

Type Package

Title Functional Tolerance Bands

Version 1.0

Date 2016-05-12

Author Lasitha Rathnayake, Pankaj Choudhary

Maintainer Lasitha Rathnayake <nir.rathnayake@gmail.com>

Description Construct tolerance bands for functional data

License GPL-3

LazyLoad tab yes

R topics documented:

`fTB-package` *Functional Tolerance Bands*

Description

Methods to construct tolerance bands for functional data.

Author(s)

Lasitha Rathnayake, Pankaj Choudhary

Maintainer: Lasitha Rathnayake

References

Rathnayake, L. N. and Choudhary, P. K. (2015) Tolerance bands for functional data , Biometrics, doi: 10.1111/biom.12434.

boot.tolfactor *bootstrap tolerance factor*

Description

Calculate tolerance factor using bootstraps

Usage

```
boot.tolfactor(n.boot, y, y.mean, y.sigma, y.G, root.int, prob, alpha, simul,
I, D, argvals)
```

Arguments

n.boot	number of bootstrap samples
y	sample from original data
y.mean	mean of the sample
y.sigma	standard deviation of the sample
y.G	standard deviation of true values
root.int	Interval for root finding
prob	content probability of the tolerance interval
alpha	
simul	TRUE (if simultaneous tolerance factor need to be computed)
I	number of subjects
D	number of points in the union grid
argvals	arguments values for gam model

Value

list of point wise critical points and simultaneous critical points for onesided upper TI .

onesided.tolfactor *Onesided tolerance factor*

Description

Calculate onesided tolerance factor for functional data

Usage

```
onesided.tolfactor(prob, alpha, y.mean, y.sigma, y.G, ystar.mean, ystar.sigma,
ystar.G, root.int, simul, I, D, n.boot)
```

Arguments

prob	content probability of tolerance intervals
alpha	1-\alpha confidence level
y.mean	mean of the sample
y.sigma	standard deviation of the sample
y.G	standard deviation of the true values
ystar.mean	mean of bootstrap
ystar.sigma	standard deviation of the sample
ystar.G	standard deviation of the true values
root.int	Interval for root finding
simul	TRUE (if simultaneous tolerance factor is computed)
I	number of subject
D	number of points in the union grid
n.boot	number of bootstraps

Value

`pwise.obs`

pointwise tolerance factor for observed curves

`sim.obs`

simultaneous tolerance factor for observed curves pwise.true pointwise tolerance factor for true curves \item sim.true simultaneous tolerance factor for true curves

---

`param.estimate` *Estimate parameters*

Description

Function for estimate parameters (mu and sigma squared and G squared and tau squared)

Usage

```
param.estimate(Y, I, D, argvals, nbasis = 10, pve = 0.99)
```

Arguments

Y	data
I	number of subjects
D	number of points in the union grid
argvals	argument values for the gam model
nbasis	number of basis functions for the gam model
pve	proportion of variance explained by the extracted principal components

Value

`mean.hat`	estimated mean
`sigma.sq.hat`	estimated sigma squared
`G.sq.hat`	estimated G squared

Note

This function is a modified version of ccb.fpc function in the refund package.

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (Y, I, D, argvals, nbasis = 10, pve = 0.99)
{
    library(mgcv)
    d.vec <- rep(argvals, each = I)
    gam0 <- gam((as.vector(Y) ~ s(d.vec, k = nbasis)), method = "REML")
    mu <- predict(gam0, newdata = data.frame(d.vec = argvals))
    Y.tilde <- Y - matrix(mu, I, D, byrow = TRUE)
    cov.sum <- cov.count <- cov.mean <- matrix(0, D, D)
    for (i in 1:I) {
        obs.points <- which(!is.na(Y[i, ]))
        cov.count[obs.points, obs.points] <- cov.count[obs.points,
            obs.points] + 1
        cov.sum[obs.points, obs.points] <- cov.sum[obs.points,
            obs.points] + tcrossprod(Y.tilde[i, obs.points])
    }
    G.0 <- ifelse(cov.count == 0, NA, cov.sum/cov.count)
    diag.G0 <- diag(G.0)
    diag(G.0) <- NA
    row.vec <- rep(argvals, each = D)
    col.vec <- rep(argvals, D)
    npc.0 <- matrix(predict(gam(as.vector(G.0) ~ te(row.vec,
        col.vec, k = nbasis), method = "REML", weights = as.vector(cov.count)),
        newdata = data.frame(row.vec = row.vec, col.vec = col.vec)),
        D, D)
    npc.0 <- (npc.0 + t(npc.0))/2
    evalues <- eigen(npc.0, symmetric = TRUE, only.values = TRUE)$values
    evalues <- replace(evalues, which(evalues <= 0), 0)
    npc <- min(which(cumsum(evalues)/sum(evalues) > pve))
    efunctions <- matrix(eigen(npc.0, symmetric = TRUE)$vectors[,
        seq(len = npc)], nrow = D, ncol = npc)
    evalues <- eigen(npc.0, symmetric = TRUE, only.values = TRUE)$values[1:npc]
    cov.hat <- efunctions %*% tcrossprod(diag(evalues, nrow = npc,
        ncol = npc), efunctions)
    DIAG <- (diag.G0 - diag(cov.hat))[floor(D * 0.2):ceiling(D *
        0.8)]
    tau.sq <- max(mean(DIAG, na.rm = TRUE), 0)
    sigma.sq <- diag(cov.hat) + tau.sq
    return.obj <- list(mean.hat = mu, sigma.sq.hat = sigma.sq,
        G.sq.hat = diag(cov.hat))
    return(return.obj)
```

```
}
```

`solve.prob.eqn` *solution of a probability equation*

Description

Function to find the solution of a probability equation

Usage

```
solve.prob.eqn(guess.root, prob.eqn, step.size = 0.2, delta = 0.1, grid.size = 0
```

Arguments

`guess.root`	temporary root for guessing an initial interval
`prob.eqn`	probability equation
`step.size`	size of the grid where the search interval is partitioned
`delta`	increment/decrement size for finding a temporary interval
`grid.size`	size of the grid where the search interval is partitioned

Value

solution of the probability equation

`TB.calc` *Calculation of tolerance bands*

Description

This function calculate tolerance bands for given functional data. It include poinwise, simultaneous bands for both true and observed curves.

Usage

```
TB.calc(y, alpha = 0.05, prob, simul = TRUE, n.boot = 500)
```

Arguments

`y`	functional data that we need to construct tolerance bands for
`alpha`	parameter for confidence level such that 1-alpha = confidence level (default = .05).
`prob`	content probability
`simul`	logical argument to indicate simultaneous tolerance bands is constructed or not.
`n.boot`	number of bootstrap

Value

Return pointwise or simultaneous tolerance intervals depending upon the arguments

`onesided.pwise TB`
: Onesided pointwise tolerance band

`twosided.pwise TB`
: Twosided pointwise tolernace band

`onesided.sim TB`
: Onesided simultaneous tolerance bands

`twosided.im TB`
: Twosided simultaneous tolerance bands

`estimated parameters`

Note

further notes

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (y, alpha = 0.05, prob, simul = TRUE, n.boot = 500)
{
    I <- dim(y)[1]
    D <- dim(y)[2]
    col.names <- as.integer(colnames(y))
    arg <- seq(col.names[1], col.names[length(col.names)], 1)
    root.int <- c(-20, 20)
    set.seed(1234)
    require(mgcv)
    y.param <- param.estimate(Y = y, I = I, D = D, argvals = arg)
    y.mean <- y.param$mean.hat
    y.sigma <- sqrt(y.param$sigma.sq.hat)
    y.G <- sqrt(y.param$G.sq.hat)
    k.boot <- boot.tolfactor(n.boot = n.boot, y = y, y.mean = y.mean,
        y.sigma = y.sigma, y.G = y.G, root.int = root.int, prob = prob,
        alpha = alpha, simul = simul, I = I, D = D, argvals = arg)
    onesided.boot.obs.u <- y.mean + k.boot[[1]]$pwise.obs * y.sigma
    onesided.boot.true.u <- y.mean + k.boot[[1]]$pwise.true *
        y.G
    onesided.pwise <- cbind(onesided.boot.obs.u, onesided.boot.true.u)
    if (simul == T) {
        onesided.boot.obs.sim <- y.mean + k.boot[[1]]$sim.obs *
            y.sigma
        onesided.boot.true.sim <- y.mean + k.boot[[1]]$sim.true *
            y.G
        onesided.sim <- cbind(onesided.boot.obs.sim, onesided.boot.true.sim)
    }
    twosided.boot.obs.u <- y.mean + k.boot[[2]]$pwise.obs * y.sigma
    twosided.boot.true.u <- y.mean + k.boot[[2]]$pwise.true *
        y.G
```

```
    twosided.boot.obs.l <- y.mean - k.boot[[2]]$pwise.obs * y.sigma
    twosided.boot.true.l <- y.mean - k.boot[[2]]$pwise.true *
        y.G
    twosided.pwise <- cbind(twosided.boot.obs.l, twosided.boot.obs.u,
        twosided.boot.true.l, twosided.boot.true.u)
    if (simul == T) {
        twosided.boot.obs.sim.u <- y.mean + k.boot[[2]]$sim.obs *
            y.sigma
        twosided.boot.obs.sim.l <- y.mean - k.boot[[2]]$sim.obs *
            y.sigma
        twosided.boot.true.sim.u <- y.mean + k.boot[[2]]$sim.true *
            y.G
        twosided.boot.true.sim.l <- y.mean - k.boot[[2]]$sim.true *
            y.G
        twosided.sim <- cbind(twosided.boot.obs.sim.l, twosided.boot.obs.sim.u,
            twosided.boot.true.sim.l, twosided.boot.true.sim.u)
    }
    if (simul == T) {
        result <- list(onesided.pwise, onesided.sim, twosided.pwise,
            twosided.sim, cbind(y.mean, y.sigma, y.G))
        names(result) <- c("onesided.pwise TB", "onesided.sim TB",
            "twosided.pwise TB", "twosided.sim TB", "estimated parameters")
    }
    else {
        result <- list(onesided.pwise, twosided.pwise, cbind(y.mean,
            y.sigma, y.G))
        names(result) <- c("onesided.pwise TB", "twosided.pwise TB",
            "estimated parameters")
    }
    return(result)
}
```

`twosided.tolfactor` *twosided tolerance factor*

Description

Calculate the twosided tolerance factor

Usage

```
twosided.tolfactor(prob, alpha, y.mean, y.sigma, y.G, ystar.mean, ystar.sigma,
ystar.G, root.int, simul, I, D, n.boot)
```

Arguments

`prob`	content probability
`alpha`	parameter for confidence level such that 1-alpha = confidence level (default = .05)
`y.mean`	mean of the sample
`y.sigma`	standard deviation of the sample
`y.G`	standard deviation of true curves

ystar.mean	bootstrap mean
ystar.sigma	boostrap standard deviation
ystar.G	boostrap standard deviation of true curves
root.int	interval for root finding
simul	TRUE (if simultaneous tolerance factor is computed)
I	number of subjects
D	number of points in the union grid
n.boot	number of bootstraps

Value

pwise.obs	pointwise tolerance factors for observed curves
sim.obs	simultaneous tolerance factors for observed curves
pwise.true	pointwise tolerance factors for true curves
sim.true	simultaneous tolerance factors for true curves