Package ‘FPV’

October 12, 2022

Type Package
Title Testing Hypotheses via Fuzzy P-Value in Fuzzy Environment
Version 0.5
Date 2017-08-21
Author Abbas Parchami (Department of Statistics, Faculty of Mathematics and Computer, Shahid Bahonar University of Kerman, Kerman, Iran)
Maintainer Abbas Parchami <parchami@uk.ac.ir>

Description The main goal of this package is drawing the membership function of the fuzzy p-value which is defined as a fuzzy set on the unit interval for three following problems: (1) testing crisp hypotheses based on fuzzy data, see Filzmoser and Viertl (2004) <doi:10.1007/s001840300269>, (2) testing fuzzy hypotheses based on crisp data, see Parchami et al. (2010) <doi:10.1007/s00362-008-0133-4>, and (3) testing fuzzy hypotheses based on fuzzy data, see Parchami et al. (2012) <doi:10.1007/s00362-010-0353-2>. In all cases, the fuzziness of data or / and the fuzziness of the boundary of null fuzzy hypothesis transported via the p-value function and causes to produce the fuzzy p-value. If the p-value is fuzzy, it is more appropriate to consider a fuzzy significance level for the problem. Therefore, the comparison of the fuzzy p-value and the fuzzy significance level is evaluated by a fuzzy ranking method in this package.

License LGPL (>= 3)
Imports FuzzyNumbers, FuzzyNumbers.Ext.2
NeedsCompilation no
Repository CRAN
Date/Publication 2017-08-29 12:57:59 UTC

R topics documented:

   FPV-package ................................................................. 2
   fuzzy.p.value ............................................................ 3

Index 10
Description

Statistical testing hypotheses has an important rule for making decision in practical and applied problems. In traditional testing methods, data, parameters, hypotheses and other elements of problem are considered crisp. But in applied sciences such as economics, agriculture and social sciences, it may be confront with vague definitions and fuzzy concepts. In such situations, the classical methods can not solve the vague test and they need to generalize for using in fuzzy environments. The vagueness entrance in testing hypotheses problem can be done via data or hypotheses. Therefore, the following three major problems can be usually considered for a fuzzy environment: (1) testing crisp hypotheses based on fuzzy data, (2) testing fuzzy hypotheses based on crisp data, and (3) testing fuzzy hypotheses based on fuzzy data. Similar to the classical testing hypotheses, one can consider different procedure methods for solving the above mentioned problems such as Neyman-Pearson, Bayes, likelihood ratio, minimax and p-value. Computing Fuzzy p-Value package, i.e. Fuzzy.p.value package, is an open source (LGPL 3) package for R which investigate on the above three problems on the basis of fuzzy p-value approach. All formulas and given examples are match with (Parchami and Mashinchi, 2016) to easily show the performance of the proposed methods.

Author(s)

Abbas Parchami (Department of Statistics, Faculty of Mathematics and Computer, Shahid Bahonar University of Kerman, Kerman, Iran)

Maintainer: Abbas Parchami <parchami@uk.ac.ir>

References

Gagolewski, M., Caha, J. (2015) A guide to the FuzzyNumbers package for R (FuzzyNumbers version 0.4-1) http://FuzzyNumbers.rexamine.com
fuzzy.p.value

See Also

FuzzyNumbers FuzzyNumbers.Ext.2 Fuzzy.p.value

fuzzy.p.value Testing hypotheses based on fuzzy hypotheses and fuzzy data: A fuzzy p-value approach

Description

Function fuzzy.p.value can draw the membership function of the fuzzy p-value for the following three major problems which can be usually considered for the following tests in a fuzzy environment: (1) testing crisp hypotheses based on fuzzy data, (2) testing fuzzy hypotheses based on crisp data, and (3) testing fuzzy hypotheses based on fuzzy data. Also, one can consider a fuzzy significance of level for test by function fuzzy.p.value.

Usage

fuzzy.p.value(t, H0b, sig = 0.05, p.value, knot.n=10, fig=c("1", "2", "3"), ...)

Arguments

  t the observed value of the test statistic. t can take one of the following precise / non-precise values:
      (1) crisp (real) value,
      (2) triangular fuzzy number using function TriangularFuzzyNumber from package FuzzyNumbers,
      (3) trapezoidal fuzzy number using function TrapezoidalFuzzyNumber from package FuzzyNumbers,
      (4) fuzzy number using function FuzzyNumber from package FuzzyNumbers, and
      (5) power fuzzy number using function PowerFuzzyNumber from package FuzzyNumbers. More details about these functions are presented in (Gagolewski and Caha, 2015).

  H0b the boundary of the null hypothesis. H0b can take one of the following precise / non-precise values:
      (1) crisp (real) value,
      (2) triangular fuzzy number using function TriangularFuzzyNumber from package FuzzyNumbers,
      (3) trapezoidal fuzzy number using function TrapezoidalFuzzyNumber from package FuzzyNumbers,
      (4) fuzzy number using function FuzzyNumber from package FuzzyNumbers, and
      (5) power fuzzy number using function PowerFuzzyNumber from package FuzzyNumbers.
**sig**

the significance level of the test with default `sig = 0.05`. `sig` can take one of the following precise / non-precise values:

1. crisp (real) value,
2. triangular fuzzy number using function `TriangularFuzzyNumber` from package `FuzzyNumbers`,
3. trapezoidal fuzzy number using function `TrapezoidalFuzzyNumber` from package `FuzzyNumbers`,
4. fuzzy number using function `FuzzyNumber` from package `FuzzyNumbers`, and
5. power fuzzy number using function `PowerFuzzyNumber` from package `FuzzyNumbers`.

More details about these functions are presented in (Gagolewski and Caha, 2015).

**p.value**

the p-value of test in non-fuzzy environment which is a function from `t` and `H0b`

**knot.n**

the number of knots with default `knot.n = 10`; see package `FuzzyNumbers`

**fig**

a numeric argument which can take only values 1, 2 or 3.

If `fig = 1`, then just the membership function of fuzzy p-value will be shown in figure.

If `fig = 2`, then the membership functions of fuzzy p-value and fuzzy significance level will be shown in a figure.

If `fig = 3`, then three membership functions of `t`, `H0b` (inputted fuzzy numbers) and also the fuzzy p-value (outputted fuzzy number) are drawn in a figure.

... additional arguments passed from `plot`

### Value

This function returns some information about the fuzzy p-value and also plot a figure for it.

**result**

returns the result of the test, i.e. returns the accepted hypothesis and also the acceptance degree of the accepted hypothesis

**cuts**

returns the `α`-cuts of the computed fuzzy p-value

**core**

returns the core of the computed fuzzy p-value

**support**

returns the support of the computed fuzzy p-value

**Delta_PS**

returns a numeric value which is need for computing `D(P > S)` and `D(S > P)`. For more details, see \(\Delta_{PS}\) from (Parchami et al., 2012)

**Delta_SP**

returns a numeric value which is need for computing `D(P > S)` and `D(S > P)`. For more details, see \(\Delta_{SP}\) from (Parchami et al., 2012)

**Degree_P_biger_than_S**

returns a real number between zero and one which show the degree of believe to the sentence "fuzzy p-value is bigger than fuzzy significance level". For more details, see \(D(P > S)\) in (Parchami et al., 2012)

**Degree_S_biger_than_P**

returns a real number between zero and one which show the degree of believe to the sentence "fuzzy significance level is bigger than fuzzy p-value". For more details, see \(D(P > S)\) in (Parchami et al., 2012)
fuzzy.p.value

accepted_hypothesis
returns the accepted hypothesis in the test. Returns "H0" iff the null hypothesis accepted, and returns "H1" iff the alternative hypothesis accepted

acceptance_degree
returns only the degree of acceptance for the accepted hypothesis in the test

Author(s)
Abbas Parchami

See Also
FuzzyNumbers FuzzyNumbers.Ext.2 Fuzzy.p.value

Examples

library(FuzzyNumbers)
library(FuzzyNumbers.Ext.2)

# Example 1:
t <- FuzzyNumber(-0.5, .6, .8, 1,
    lower=function(alpha) qbeta(alpha,0.4,3),
    upper=function(alpha) (1-alpha)^4
 )
H0b = PowerFuzzyNumber(.5,1.2,1.5,1.6, p.left=1, p.right=0.5)
p.value = function(t,H0b) pt((t-H0b)/sqrt(1/9), df=8)
fuzzy.p.value(t, H0b, sig=.05, p.value, knot.n=20, fig=1, lty=4, lwd=4, col=6)
fuzzy.p.value(t, H0b, sig=.05, p.value, knot.n=20, fig=2)$result
sig = TriangularFuzzyNumber(0, .03, .30)
fuzzy.p.value(t, H0b, sig, p.value, knot.n=20, fig=2)$cuts #Only print alpha-cuts of fuzzy p-value

sig = TrapezoidalFuzzyNumber(0, .05, .05, .20)
fuzzy.p.value(t, H0b, sig, p.value, knot.n=20, fig=3, col=2)$accepted

# Example 2: For working by some elements of fuzzy p-value (continue of Example 1)
Fuzzy.p.value <- fuzzy.p.value(t, H0b, sig=.05, fig=1, p.value, knot.n=4)
class(Fuzzy.p.value)
print( Fuzzy.p.value )

Fuzzy.p.value$core #Core of fuzzy p-value
Fuzzy.p.value$support #Support of fuzzy p-value

# Upper bounds of fuzzy p-value
Fuzzy.p.value$cuts[,"U"] #Or equivalently, Fuzzy.p.value$cuts[,2]

# Example 3: (Exam 4.4 from persian p-value paper)
knot.n = 10
t = TriangularFuzzyNumber(1315, 1327, 1342)
H0b = TriangularFuzzyNumber(1275, 1300, 1325)
sig = TriangularFuzzyNumber(0, .05, .1)
p.value = function(t,H0b) 1-pnorm((t-H0b)/(120/6))
fuzzy.p.value(t, H0b, sig, p.value, knot.n, fig=3)

# Example 4: (Exam 4.5 from persian p-value paper, where X~P(12*lambda) )
knot.n = 200
t = TriangularFuzzyNumber(24, 27, 30)
H0b = TriangularFuzzyNumber(2.75, 3.25, 3.25)
sig = TriangularFuzzyNumber(0, .05, .1)
p.value = function(t,H0b) ppois(t, 12*H0b)
fuzzy.p.value(t, H0b, sig, p.value=p.value, knot.n, fig=2, lwd=2)
# Repeat example with knot.n=10 to give a non-precise result

# Example 5: A new example
t <- FuzzyNumber(1, 1.4, 1.8, 2,
    lower=function(alpha) pbeta(alpha,2,1),
    upper=function(alpha) 1-sqrt(alpha))
H0b = TriangularFuzzyNumber(4,5,7)
p.value = function(t,H0b) pt( (t-H0b)/sqrt(1/4), df=4)
fuzzy.p.value(t, H0b, sig=.1^3, p.value, knot.n=10, fig=3, col=2, lwd=2, xlim=c(0,.012))

# ---------- Examples of Springer fuzzy p-value paper ------------------

# Example 1 (Springer fuzzy p-value).
T1 = TriangularFuzzyNumber(1257,1261,1278)
T2 = TriangularFuzzyNumber(1251,1287,1302)
T3 = TriangularFuzzyNumber(1315,1346,1372)
T4 = TriangularFuzzyNumber(1306,1330,1348)
T5 = TriangularFuzzyNumber(1298,1329,1349)
T6 = TriangularFuzzyNumber(1288,1301,1320)
T7 = TriangularFuzzyNumber(1298,1317,1333)
T8 = TriangularFuzzyNumber(1241,1269,1284)
T9 = TriangularFuzzyNumber(1325,1353,1369)
T10= TriangularFuzzyNumber(1301,1337,1355)
t = 10^(-1)*(T1+T2+T3+T4+T5+T6+T7+T8+T9+T10)
t # T(1288,1313,1331)

plot(T1, add=FALSE, lwd=2, xlim=c(1230,1380))
plot(T2, add=TRUE, lwd=2)
plot(T3, add=TRUE, lwd=2)
plot(T4, add=TRUE, lwd=2)
plot(T5, add=TRUE, lwd=2)
plot(T6, add=TRUE, lwd=2)
plot(T7, add=TRUE, lwd=2)
plot(T8, add=TRUE, lwd=2)
plot(T9, add=TRUE, lwd=2)
plot(T10, add=TRUE, lwd=2)
plot(t, add=TRUE, col=2, lwd=4)

H0b = 1300
# T ~ N(1300,30^2/10)
p.value = function(t,H0b) pnorm( t, mean=1300, sd=30/sqrt(10), lower.tail=FALSE)
# Or equivalently
p.value = function(t,H0b) pnorm( (t-1300)/(30/sqrt(10)), lower.tail=FALSE)
sig = TriangularFuzzyNumber(0,0.05,0.1)
fuzzy.p.value(t, H0b, sig, p.value, knot.n=50, fig=2, lwd=2, xlim=c(0,1))

# Example 2. (continue of Example 1)
t = TriangularFuzzyNumber(1300,1313,1321)
p.value = function(t,H0b) 2 * pnorm( t, mean=1300, sd=30/sqrt(10), lower.tail=FALSE)
sig = TriangularFuzzyNumber(0,0.15,0.3)
fuzzy.p.value(t, H0b, sig, p.value, knot.n=50, fig=3, lwd=2)

# Example 4 (Springer fuzzy p-value) X ~ N(mu,sigma^2).
sigma =120
n = 36
x.bar <- 1327
H0b = TriangularFuzzyNumber(1275, 1300, 1325)
sig = TriangularFuzzyNumber(0, 0.15, 0.3)
p.value = function(x.bar,H0b) pnorm( x.bar, mean=H0b, sd=sigma/sqrt(n), lower.tail=FALSE)
fuzzy.p.value(x.bar, H0b, sig, p.value, knot.n=10, fig=2, lwd=2, xlim=c(0,1))

#Continue
sig1 = PowerFuzzyNumber(0, 0.15, 0.3, p.left=2, p.right=2)
plot(sig1, xlim=c(0,.6))
sig2 = PowerFuzzyNumber(0, 0.15, 0.3, p.left=.5, p.right=.5)
plot(sig2, col=2, add=TRUE)
fuzzy.p.value(x.bar, H0b, sig1, p.value, knot.n=10, fig=2, lwd=2, xlim=c(0,1))
fuzzy.p.value(x.bar, H0b, sig2, p.value, knot.n=10, fig=2, lwd=2, xlim=c(0,1))

## The function is currently defined as
function (t, H0b, sig = 0.05, p.value, knot.n = 10, fig = c("1",
"2", "3"), ...) {
  if (fig == 1) {
    P = f2apply(t, H0b, p.value, knot.n = knot.n, type = "1",
                1.0.plot = FALSE, ...
  }
fuzzy.p.value

else {
    if (fig == 2) {
        P = f2apply(t, H0b, p.value, knot.n = knot.n, type = "l",
                   I.O.plot = FALSE, ...)
        if (class(sig) == "numeric") {
            abline(v = sig, col = 2, lty = 3)
        } else {
            plot(sig, col = 2, lty = 3, add = TRUE)
        }
        legend("topright", c("Fuzzy p-value", "Significance level"),
               col = c(1, 2), text.col = 1, lwd = c(1, 1), lty = c(1, 3),
               bty = "n")
    } else {
        if (fig == 3) {
            P = f2apply(t, H0b, p.value, knot.n = knot.n,
                        type = "l", I.O.plot = TRUE, ...)
            x = t
            y = H0b
        }
    }
    if (class(sig) == "numeric") {
        sig <- TriangularFuzzyNumber(sig, sig, sig)
    }
    P_L = P$cuts[, "L"]
    P_L = P_L[length(P_L):1]
    P_U = P$cuts[, "U"]
    P_U = P_U[length(P_U):1]
    S_L = alphacut(sig, round(seq(0, 1, len = knot.n), 5))[,
                  "L"]
    S_U = alphacut(sig, round(seq(0, 1, len = knot.n), 5))[,
                  "U"]
    Int1 = (P_U - S_L) * (P_U > S_L)
    Int2 = (P_L - S_U) * (P_L > S_U)
    Arz = 1/(knot.n - 1)
    Integral1 <- (sum(Int1) - Int1[1]/2 - Int1[length(Int1)]/2) * Arz
    Integral2 <- (sum(Int2) - Int2[1]/2 - Int2[length(Int2)]/2) * Arz
    Delta_PS = Integral1 + Integral2
    Int3 = (S_U - P_L) * (S_U > P_L)
    Int4 = (S_L - P_U) * (S_L > P_U)
    Integral3 <- (sum(Int3) - Int3[1]/2 - Int3[length(Int3)]/2) * Arz
    Integral4 <- (sum(Int4) - Int4[1]/2 - Int4[length(Int4)]/2) * Arz
    Delta_SP = Integral3 + Integral4
    Degree_P_biger_than_S = Delta_PS/(Delta_PS + Delta_SP)
    Degree_S_biger_than_P = 1 - Degree_P_biger_than_S
    if (Degree_P_biger_than_S >= Degree_S_biger_than_P) {
        result = noquote(paste("The null hypothesis (H0) is accepted with degree D(P>S)=",
                                 Degree_P_biger_than_S))
    }
}
fuzzy.p.value

round(Degree_P_biger_than_S, 4), ", at the considered significance level."))
accepted_hypothesis = noquote("H0")
acceptance_degree = Degree_P_biger_than_S
}
else {
    if (Degree_P_biger_than_S < Degree_S_biger_than_P) {
        result = noquote(paste("The alternative hypothesis (H1) is accepted with degree D(S>P)=", 
                               round(Degree_S_biger_than_P, 4), ", at the considered significance level."))
        accepted_hypothesis = noquote("H1")
        acceptance_degree = Degree_S_biger_than_P
    }
    else {
        print(noquote(paste0("Impossible case")))
    }
}
return(list(result = result, cuts = P$cuts, core = P$core, 
support = P$support, Delta_PS = as.numeric(Delta_PS), 
Delta_SP = as.numeric(Delta_SP), Degree_P_biger_than_S = as.numeric(Degree_P_biger_than_S), 
Degree_S_biger_than_P = as.numeric(Degree_S_biger_than_P), 
accepted_hypothesis = accepted_hypothesis, acceptance_degree = as.numeric(acceptance_degree)))
}
Index

* Fuzzy data
  FPV-package, 2
  fuzzy.p.value, 3
* Fuzzy hypothesis
  FPV-package, 2
  fuzzy.p.value, 3
* Fuzzy p-value
  FPV-package, 2
  fuzzy.p.value, 3
* Fuzzy ranking method
  FPV-package, 2
  fuzzy.p.value, 3
* Fuzzy significance level
  FPV-package, 2
  fuzzy.p.value, 3
* FuzzyNumber
  FPV-package, 2
  fuzzy.p.value, 3
* PowerFuzzyNumber
  FPV-package, 2
  fuzzy.p.value, 3
* Testing hypotheses
  FPV-package, 2
  fuzzy.p.value, 3
* TrapezoidalFuzzyNumber
  FPV-package, 2
  fuzzy.p.value, 3
* TriangularFuzzyNumber
  FPV-package, 2
  fuzzy.p.value, 3

FPV (FPV-package), 2
FPV-package, 2
fuzzy.p.value, 3