# Package 'twostageTE'

October 14, 2022

**Type** Package

**Title** Two-Stage Threshold Estimation

**Version** 1.3

**Date** 2015-08-29

**Author** Shawn Mankad, George Michailidis, Moulinath Banerjee

**Maintainer** Shawn Mankad <smankad@cornell.edu>

**Description** Implements a variety of non-parametric methods for computing one-stage and two-stage confidence intervals, as well as point estimates of threshold values.

**License** GPL-2

**Depends** isotone

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-09-27 18:09:43

## R topics documented:

---

`chernoff_realizations` *Quantiles of the Chernoff Random Variable*

---

## Description

Quantiles of the Chernoff Random Variable that are used within the wald-type confidence interval functions.

## Usage

```
data(chernoff_realizations)
```

## Format

A data frame of length 200 with quantiles and density values.

## References

Computing Chernoff's Distribution Piet Groeneboom, Jon A Wellner Journal of Computational and Graphical Statistics Vol. 10, Iss. 2, 2001

## Examples

```
data(chernoff_realizations)
```

---

`estimateDeriv` *Derivative Estimation*

---

## Description

Estimate derivative of a function at a point d_0 based on a local quadratic regression procedure of Fan and Gijbels (1996) that utilizes an automatic bandwidth selection formula.

## Usage

```
estimateDeriv(explanatory, response, d_0, sigmaSq)
```

## Arguments

| | |
|---|---|
| `explanatory` | Explanatory sample points |
| `response` | Observed responses at the explanatory sample points |
| `d_0` | d_0 is the point of interest where the derivative is estimated |
| `sigmaSq` | estimate of variance at d_0 |

## Details

This is an internal function not meant to be called directly.

## Value

Returns a single number representing the derivative estimate at d_0. If a negative derivative has been estimated, then a warning is given, as this violates the isotonic (non-decreasing) assumption.

## Author(s)

Shawn Mankad

## References

Fan J, Gijbels I (1996). Local polynomial modelling and its applications, volume 66 of Monographs on Statistics and Applied Probability. Chapman & Hall, London. ISBN 0-412-98321-4.

## Examples

```
explanatory = runif(50)
response = explanatory^2 + rnorm(50, sd=0.1)
estimateDeriv(explanatory, response, d_0=0.5,
    sigmaSq=estimateSigmaSq(explanatory, response)$sigmaSq)

## The function is currently defined as
function (explanatory, response, d_0, sigmaSq)
{
    deriv_estimateHelper <- function(explanatory, response, d_0,
        sigmaSq) {
        n = length(response)
        p = 5
        X = matrix(0, n, p)
        for (i in 1:p) {
            X[, i] = (explanatory - d_0)^i
        }
        beta_hat = lm(response ~ 0 + X)$coef
        h = 0
        for (i in (p - 1):(p + 1)) {
            j = i - p + 2
            h = h + beta_hat[i - 1] * factorial(j) * d_0^(j -
                1)
        }
        return(2.275 * (sigmaSq/h^2)^(1/7) * n^(-1/7))
    }
    n = length(response)
    p = 2
    X = matrix(0, n, p)
    X[, 1] = (explanatory - d_0)
    X[, 2] = (explanatory - d_0)^2
    bw_opt = deriv_estimateHelper(explanatory, response, d_0,
        sigmaSq)
    W = 0.75/bw_opt * sapply(1 - ((explanatory - d_0)/bw_opt)^2,
```

```
        max, 0)
    while (sum(W > 1) <= 1 & bw_opt <= max(explanatory) - min(explanatory)) {
        bw_opt = bw_opt * 2
        W = 0.75/bw_opt * sapply(1 - ((explanatory - d_0)/bw_opt)^2,
            max, 0)
    }
    beta_hat = lm(response ~ 0 + X, weight = W)$coef
    while (beta_hat[1] <= 0 & bw_opt <= max(explanatory) - min(explanatory)) {
        bw_opt = bw_opt * 2
        W = 0.75/bw_opt * sapply(1 - ((explanatory - d_0)/bw_opt)^2,
            max, 0)
        beta_hat = lm(response ~ 0 + X, weight = W)$coef
    }
    if (beta_hat[1] <= 0) {
        warning("deriv_estimate:WARNING: NEGATIVE DERIVATIVE HAS BEEN ESTIMATED",
            .call = FALSE)
        return(1/log(n))
    }
    return(beta_hat[1])
}
```

---

`estimateSigmaSq` *Estimate Variance*

---

## Description

Estimate variance using Gasser, Sroka, and Jennen-Steinmetz, 1986

## Usage

```
estimateSigmaSq(explanatory, response)
```

## Arguments

| | |
|---|---|
| `explanatory` | Explanatory sample points |
| `response` | Observed responses at the explanatory sample points |

## Value

Returns a list consisting of

| | |
|---|---|
| `sigmaSq` | Estimate of variance |
| `a` | coefficients of the estimator |
| `b` | coefficients of the estimator |
| `eps` | coefficients of the estimator |

## Author(s)

Shawn Mankad

## References

Gasser T, Sroka L, Jennen-Steinmetz C (1986). 'Residual variance and residual pattern in nonlinear regression.' Biometrika, 73(3), 625-633. ISSN 0006-3444.

## Examples

```
explanatory = runif(50)
response = explanatory^2 + rnorm(50, sd=0.1)
estimateSigmaSq(explanatory, response)

## The function is currently defined as
function (explanatory, response)
{
    ind = order(explanatory, decreasing = FALSE)
    if (sum(diff(ind) < 0) != 0) {
        explanatory = explanatory[ind]
        response = response[ind]
    }
    n = length(response)
    a = b = eps = rep(0, n - 2)
    for (i in 2:(n - 1)) {
        x = explanatory[(i - 1):(i + 1)]
        a[i - 1] = (x[3] - x[2])/(x[3] - x[1])
        b[i - 1] = (x[2] - x[1])/(x[3] - x[1])
        eps[i - 1] = a[i - 1] * response[i - 1] + b[i - 1] *
            response[i + 1] - response[i]
    }
    cSq = 1/(a^2 + b^2 + 1)
    list(sigmaSq = 1/(n - 2) * sum(cSq * eps^2), a = a, b = b,
        eps = eps)
  }
```

---

# likelihoodConfidenceInterval

*Likelihood ratio based confidence intervals*

---

## Description

This is an internal function not meant to be called directly. Inverts the likelihood ratio statistic to form confidence intervals.

## Usage

```
likelihoodConfidenceInterval(explanatory, response, Y_0, level = NA)
```

## Arguments

| | |
|---|---|
| explanatory | Explanatory sample points |
| response | Observed responses at the explanatory sample points |
| Y_0 | Threshold of interest |
| level | Desired confidence level for the confidence interval |

## Value

Returns a list with

| | |
|---|---|
| estimate | Threshold estimate |
| lower | Lower bound of the confidence interval |
| upper | Upper bound of the confidence interval |
| sigmaSq | Estimate of variance |
| deriv_d0 | Value of NA since this is not estimated. |

## Author(s)

Shawn Mankad

## Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
oneStage_LR=likelihoodConfidenceInterval(X, Y, 0.25, 0.95)

## The function is currently defined as
function (explanatory, response, Y_0, level = NA)
{
    if (is.na(level))
        level = 0.95
    RVforLR_realizations <- NULL; rm(RVforLR_realizations); # Dummy to trick R CMD check
    data("RVforLR_realizations", envir =environment())
    D = quantile(RVforLR_realizations, level)
    n = length(response)
    ind = order(explanatory, decreasing=FALSE)
    if (sum(diff(ind) < 0) != 0) {
explanatory = explanatory[ind]
response = response[ind]
    }
    fit = threshold_estimate_ir(explanatory, response, Y_0)
    sigmaSq = estimateSigmaSq(explanatory, response)$sigmaSq
    likelihoodRatio <- function(explanatory, response, X_0, Y_0,
        sigmaSq) {
        logLikelihood <- function(Y, Y_hat) {
            -1/(2 * sigmaSq) * sum((Y - Y_hat)^2)
        }
        unconstrainedLikelihood <- function(explanatory, response) {
            fit = pava(explanatory, response)
```

```
            tmp = logLikelihood(fit$response_obs, fit$y)
            return(list(x = fit$x, y_hat = fit$y, y = fit$response_obs,
                logLikelihood = tmp))
        }
        constrainedLikelihood <- function(explanatory, response,
            X_0, Y_0) {
            fit = pava(explanatory, response, X_0, Y_0)
            tmp = logLikelihood(fit$response_obs, fit$y)
            return(list(x = fit$x, y_hat = fit$y, y = fit$response_obs,
                logLikelihood = tmp))
        }
        unconst = unconstrainedLikelihood(explanatory, response)
        const = constrainedLikelihood(explanatory, response,
            X_0, Y_0)
        return(unconst$logLikelihood - const$logLikelihood)
    }
    i = fit$index + 1
    lrt_tmp = 0
    while (lrt_tmp < D && i < n) {
        lrt_tmp = likelihoodRatio(explanatory, response, explanatory[i],
            Y_0, sigmaSq)
        i = i + 1
    }
    right = explanatory[min(i, n)]
    i = fit$index - 1
    lrt_tmp = 0
    while (lrt_tmp < D && i > 1) {
        lrt_tmp = likelihoodRatio(explanatory, response, explanatory[i],
            Y_0, sigmaSq)
        i = i - 1
    }
    left = explanatory[max(i, 1)]
    return(list(estimate = fit$threshold_estimate_explanatory,
        lower = left, upper = right, sigmaSq = sigmaSq, deriv_d0 = NA))
}
```

---

`linearBootstrapConfidenceInterval_stageTwo` *Confidence interval based on bootstrapping a local linear model*

---

## Description

Implements the two stage local linear bootstrapping procedure in Tang et al. (2011)

## Usage

```
linearBootstrapConfidenceInterval_stageTwo(explanatory, response,
    Y_0, level = NA)
```

## Arguments

| | |
|---|---|
| explanatory | Explanatory sample points |
| response | Observed responses at the explanatory sample points |
| Y_0 | Threshold of interest |
| level | confidence level for the confidence interval (defaults to 0.95) |

## Value

Returns a list with

| | |
|---|---|
| estimate | threshold estimate |
| lower | Lower bound of the confidence interval |
| upper | Upper bound of the confidence interval |
| sigmaSq | Estimate of the variance |
| deriv_d0 | Value of NA since this is not estimated. |

## Author(s)

Shawn Mankad

## References

Tang R, Banerjee M, Michailidis G (2011). 'A two-stage hybrid procedure for estimating an inverse regression function.' The Annals of Statistics, 39, 956-989.

## Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
oneStage_IR=stageOneAnalysis(X, Y, 0.25, type="IR-wald", 0.99)
X2 = c(rep(oneStage_IR$L1,37),rep(oneStage_IR$U1,38))
Y2=X2^2+rnorm(n=length(X2), sd=0.1)
twoStage_IR_locLinear=likelihoodConfidenceInterval(X, Y, 0.25, 0.95)

## The function is currently defined as
function (explanatory, response, Y_0, level = NA)
{
    numBootstrap = 1000
    if (is.na(level)) {
        level = 0.95
    }
    alpha = 1 - level
    n = length(response)
    fit = threshold_estimate_locLinear(explanatory, response,
        Y_0)
    Rn = rep(0, numBootstrap)
    for (i in 1:numBootstrap) {
        ind = sample(x = n, replace = TRUE)
        fit_bst = threshold_estimate_locLinear(explanatory[ind],
```

```
            response[ind], Y_0)
        Rn[i] = sqrt(n) * (fit_bst$threshold_estimate_explanatory -
            fit$threshold_estimate_explanatory)
    }
    qU = quantile(Rn, alpha/2)
    qL = quantile(Rn, level + alpha/2)
    uBand = fit$threshold_estimate_explanatory - n^(-1/2) * qU
    lBand = fit$threshold_estimate_explanatory - n^(-1/2) * qL
    return(list(estimate = fit$threshold_estimate_explanatory,
        lower = max(lBand, min(explanatory)), upper = min(uBand,
            max(explanatory)), sigmaSq = NA, deriv_d0 = NA))
}
```

---

pava *isotonic regression*

---

## Description

This is an internal function not meant to be called directly. Wrapper for gpava in package isotone to apply the pava algorithm for isotonic regression

## Usage

```
pava(explanatory, response, X_0 = NA, Y_0 = NA, w = NA)
```

## Arguments

| | |
|---|---|
| explanatory | Explanatory sample points |
| response | Observed responses at the explanatory sample points |
| X_0 | can ignore |
| Y_0 | can ignore |
| w | weights if given repeated observations at same explanatory point |

## Value

return(list(x = explanatory, y = response_fit, response_obs = response)) List with

| | |
|---|---|
| x | Explanatory sample points |
| y | estimated isotonic regression values |
| response_obs | Observed responses at the explanatory sample points |

## Author(s)

Shawn Mankad

## References

de Leeuw J, Hornik K, Mair P (2009). 'Isotone Optimization in R: Pool-Adjacent-Violators Algorithm (PAVA) and Active Set Methods.' Journal of Statistical Software, 32(5), 1-24. ISSN 1548-7660. URL http://www.jstatsoft.org/v32/i05.

## Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
pava(X, Y, 0.25, 0.5)

## The function is currently defined as
function (explanatory, response, X_0 = NA, Y_0 = NA, w = NA)
{
    require(isotone)
    if (is.na(w))
        w = rep(1, length(explanatory))
    ind = order(explanatory, decreasing = FALSE)
    if (sum(diff(ind) < 0) != 0) {
        explanatory = explanatory[ind]
        response = response[ind]
    }
    if (is.na(X_0) && is.na(Y_0)) {
        fit = gpava(explanatory, response)
        response_fit = fit$x
    }
    else if (is.na(X_0) || is.na(Y_0)) {
        warning("Only X_0 or only Y_0 was supplied. Please check arguments.")
    }
    else {
        n = length(explanatory)
        if (sum(response < Y_0) == n && sum(explanatory < X_0) ==
            n) {
            warning("Warning: X_0 and Y_0 are outside observed region")
            fit = gpava(explanatory, response)
            response_fit = fit$x
        }
        else if (sum(response < Y_0) == n && sum(explanatory <
            X_0) == 0) {
            warning("Warning: X_0 and Y_0 are outside observed region")
            return(list(x = explanatory, y = rep(Y_0, n), y_compressed = rep(Y_0,
                n)))
        }
        else if (sum(response < Y_0) == n) {
            warning("Warning: Y_0 is outside observed region")
            n2 = n - sum(explanatory < X_0)
            y1 = response[explanatory < X_0]
            x1 = explanatory[explanatory < X_0]
            fit = gpava(x1, y1)
            response_fit = c(sapply(fit$x, min, Y_0), rep(Y_0,
                n2))
        }
```

```
        else if (sum(response >= Y_0) == n && sum(explanatory <
            X_0) == n) {
            warning("Warning: X_0 and Y_0 are outside observed region")
            return(list(x = explanatory, y = rep(Y_0, n), y_compressed = rep(Y_0,
                n)))
        }
        else if (sum(response >= Y_0) == n && sum(explanatory <
            X_0) == 0) {
            warning("Warning: X_0 and Y_0 are outside observed region")
            fit = gpava(explanatory, response)
            response_fit = fit$x
        }
        else if (sum(response >= Y_0) == n) {
            warning("Warning: Y_0 is outside observed region")
            n2 = n - sum(explanatory > X_0)
            y1 = response[explanatory > X_0]
            x1 = explanatory[explanatory > X_0]
            fit = gpava(x1, y1)
            response_fit = c(rep(Y_0, n2), sapply(fit$x, max,
                Y_0))
        }
        else if (sum(explanatory < X_0) == n) {
            warning("Warning: X_0 is outside observed region")
            fit = gpava(explanatory, response)
            response_fit = sapply(fit$x, min, Y_0)
        }
        else if (sum(explanatory < X_0) == 0) {
            warning("Warning: X_0 is outside observed region")
            fit = gpava(explanatory, response)
            response_fit = sapply(fit$x, max, Y_0)
        }
        else {
            y1 = response[explanatory < X_0]
            x1 = explanatory[explanatory < X_0]
            y2 = response[explanatory >= X_0]
            x2 = explanatory[explanatory >= X_0]
            fit1 = gpava(x1, y1)
            fit2 = gpava(x2, y2)
            response_fit = c(sapply(fit1$x, min, Y_0), sapply(fit2$x,
                max, Y_0))
        }
    }
    return(list(x = explanatory, y = response_fit, response_obs = response))
}
```

---

`plot.twostageTE` *Plot function for twostageTE*

---

## Description

Plots a twostageTE object, displaying samples, point estimate and confidence interval

## Usage

```
## S3 method for class 'twostageTE'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| x | twostageTE object |
| ... | ignored |

## Value

Scatterplot of the samples and estimated regression, with confidence intervals

## Author(s)

Shawn Mankad

## Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
oneStage_IR=stageOneAnalysis(X, Y, 0.25, type="IR-wald", 0.99)
plot(oneStage_IR)

## The function is currently defined as
function (x, ...)
{
    if (!inherits(x, "twostageTE")) {
        stop("Error:  Object is not of class twostageTE")
    }
    plot_gpava <- function(x, main = "PAVA Plot", xlab = "Predictor",
        ylab = "Response", col = "lightblue", ...) {
        o <- order(x$z)
        xval <- x$z[o]
        yval <- x$x[o]
        xcum <- c(xval[1] - mean(diff(xval)), xval)
        jumps <- ((1:length(yval))[!duplicated(yval)] - 1)[-1]
        jumps <- c(1, jumps, length(xval))
        lines(xval, yval, col = col, lwd = 1, type = "S")
        points(xval[jumps], yval[jumps], col = col, pch = 13)
    }
    pava1 = gpava(z = x$X1, y = x$Y1)
    if (!is.na(x$L2)) {
        pava2 = gpava(z = x$X2, y = x$Y2)
    }
    if (!is.na(x$L2)) {
        plot(x = x$X1, y = x$Y1, pch = "1", cex = 1.5, xlab = "",
            ylab = "", ylim = range(c(x$Y1, x$Y2)), col = "grey80")
        abline(h = x$threshold, lty = 3, lwd = 1, col = 2)
        points(x = x$X2, y = x$Y2, pch = "2", cex = 1.5, col = "grey65")
        plot_gpava(pava2, col = "blue")
```

```
    }
    else {
        plot(x = x$X1, y = x$Y1, pch = "1", cex = 1.5, xlab = "",
            ylab = "", col = "grey80")
        abline(h = x$threshold, lty = 3, lwd = 1, col = 2)
        plot_gpava(pava1, col = 1)
    }
    abline(v = x$L1, lty = 2, lwd = 2)
    abline(v = x$U1, lty = 2, lwd = 2)
    if (!is.na(x$L2)) {
        abline(v = x$L2, col = "blue", lwd = 2)
        abline(v = x$U2, col = "blue", lwd = 2)
    }
    points(x = x$estimate, y = x$threshold, col = "blue", pch = 4,
        cex = 1.5)
    if (!is.na(x$L2)) {
        segments(x$estimate, min(c(x$Y1, x$Y2)) - 1, x$estimate,
            x$threshold, lwd = 2, col = "blue")
    }
    else {
        segments(x$estimate, min(x$Y1) - 1, x$estimate, x$threshold,
            lwd = 2, col = "blue")
    }
    mtext("Explanatory", side = 1, line = 2.5, cex = 1.65)
    mtext("Response", side = 2, line = 2, cex = 1.65)
    if (!is.na(x$L2)) {
        legend("topleft", c("Estimate", "1st Stage CI", "2nd Stage CI",
            "2nd Stage Iso-Regression"), pch = c(4, NA, NA, 13),
            col = c("blue", 1, "blue", "blue"), lty = c(NA, 2,
                1, 1), lwd = c(NA, 2, 2, 1), bg = "white")
    }
    else {
        legend("topleft", c("Estimate", "1st Stage CI", "1st Stage Iso-Regression"),
            pch = c(4, NA, 13), col = c("blue", 1, 1), lty = c(NA,
                2, 1), lwd = c(NA, 2, 1), bg = "white")
    }
}
```

---

`print.twostageTE` *print for twostageTE*

---

## Description

print method for twostageTE

## Usage

```
## S3 method for class 'twostageTE'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | twostageTE object |
| ... | ignored |

### Value

prints basic information about the object (point estimate and confidence intervals)

### Author(s)

Shawn Mankad

### Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
oneStage_IR=stageOneAnalysis(X, Y, 0.25, type="IR-wald", 0.99)
print(oneStage_IR)

## The function is currently defined as
function (x, ...)
{
    if (!inherits(x, "twostageTE")) {
        stop("Error:  Object is not of class twostageTE")
    }
    if (!is.null(cl <- x$call)) {
        names(cl)[2] <- ""
        cat("Call:\n")
        dput(cl)
    }
    cat(sprintf("\n%.1f%% Confidence Interval", x$level * 100))
    if (is.na(x$L2)) {
        cat(sprintf("\nn   Lower   d0_hat   Upper\n%d   %.3f   %.3f   %.3f\n",
            length(x$Y1), x$L1, x$estimate, x$U1))
    }
    else {
        cat(sprintf("\nn1   n2   Lower   d0_hat   Upper\n%d   %d   %.3f   %.3f   %.3f\n",
            length(x$Y1), length(x$Y2), x$L2, x$estimate, x$U2))
    }
    invisible(x)
  }
```

---

## RVforLR_realizations *Realizations of Random variable for LR-based confidence intervals*

---

### Description

Realizations of the random variable that the likelihood ratio test statistic converges to

### Usage

```
data("RVforLR_realizations")
```

### Format

A data frame of realizations.

### References

Banerjee M (2000). Likelihood Ratio Inference in Regular and Non-regular Problems. Ph.D. thesis, University of Washington.

Banerjee M (2009). 'Inference in exponential family regression models under certain shape constraints.' In Advances in Multivariate Statistical Methods, Statistical Science and Interdisciplinary Research, volume 4, pp. 249-72. World Scientific.

Banerjee M, Wellner J (2001). 'Likelihood Ratio Tests for Monotone Functions.' Annals of Statistics, 29, 1699 - 1731.

### Examples

```
data("RVforLR_realizations")
```

---

## stageOneAnalysis *Stage one analysis*

---

### Description

Wrapper function for twoStageTE that users can directly call on their data.

### Usage

```
stageOneAnalysis(explanatory, response, threshold,
    type = "IR-wald", level = 0.99)
```

### Arguments

| | |
|---|---|
| explanatory | Explanatory sample points |
| response | Observed responses at the explanatory sample points |
| threshold | Threshold of interest |
| type | String input of either "IR-wald" (default) or "IR-likelihood" |
| level | Desired confidence level (defaults to 0.99) |

## Value

List:

| | |
|---|---|
| L1 | Lower bound of CI |
| U1 | Upper bound of CI |
| estimate | Threshold estimate |
| level | Confidence level |
| X1 | First stage explanatory variable |
| Y1 | First stage response variable |
| X2 | NA |
| Y2 | NA |
| L2 | NA |
| U2 | NA |
| call | Method call |
| sigmaSq | Estimate of variance |
| deriv_d0 | Derivative estimate |
| class | twostageTE |

## Author(s)

Shawn Mankad

## See Also

See Also as stageTwoAnalysis, ~~~

## Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
oneStage_IR=stageOneAnalysis(X, Y, 0.25, type="IR-wald", 0.99)

## The function is currently defined as
function (explanatory, response, threshold, type = "IR-wald",
    level = 0.99)
{
    cl1 <- match.call(expand.dots = TRUE)
    if (type == "IR-wald") {
        CI = waldConfidenceInterval_ir_stageOne(explanatory,
            response, threshold, level = level)
        return(structure(list(L1 = CI$lower, U1 = CI$upper, estimate = CI$estimate,
            C_1 = CI$C_1, threshold = threshold, level = level,
            X1 = explanatory, Y1 = response, X2 = NA, Y2 = NA,
            L2 = NA, U2 = NA, call = cl1, sigmaSq = CI$sigmaSq,
            deriv_d0 = CI$deriv_d0), class = "twostageTE"))
    }
```

```
    else if (type == "IR-likelihood") {
        CI = likelihoodConfidenceInterval(explanatory, response,
            threshold, level = level)
        return(structure(list(L1 = CI$lower, U1 = CI$upper, estimate = CI$estimate,
            threshold = threshold, level = level, X1 = explanatory,
            Y1 = response, X2 = NA, Y2 = NA, L2 = NA, U2 = NA,
            call = cl1, sigmaSq = CI$sigmaSq, deriv_d0 = CI$deriv_d0),
            class = "twostageTE"))
    }
    else if (type == "SIR") {
        CI = waldConfidenceInterval_sir_stageOne(explanatory,
            response, threshold, level = level)
        return(structure(list(L1 = CI$lower, U1 = CI$upper, estimate = CI$estimate,
            threshold = threshold, level = level, X1 = explanatory,
            Y1 = response, X2 = NA, Y2 = NA, L2 = NA, U2 = NA,
            call = cl1, sigmaSq = CI$sigmaSq, deriv_d0 = CI$deriv_d0),
            class = "twostageTE"))
    }
    else error("stageOneAnalysis: type should be either 'IR-wald',
        'IR-likelihood' or 'SIR'")
}
```

---

`stageTwoAnalysis` *Stage two analysis*

---

## Description

Wrapper function for twoStageTE that users can directly call on their data.

## Usage

```
stageTwoAnalysis(stageOne, explanatory, response, type = "IR-wald",
    level = 0.95, combineData=FALSE)
```

## Arguments

| | |
|---|---|
| `stageOne` | Object returned from calling the function stageOneAnalysis |
| `explanatory` | Explanatory sample points |
| `response` | Observed responses at the explanatory sample points |
| `type` | String input of either "IR-wald" (default), "IR-likelihood" or "locLinear" |
| `level` | Confidence level (defaults to 0.95) |
| `combineData` | Optional boolean input on whether to combine data from both stages. Default is FALSE. |

## Value

List:

| | |
|---|---|
| L1 | Lower bound of CI |
| U1 | Upper bound of CI |
| estimate | Threshold estimate |
| level | Confidence level |
| X1 | First stage explanatory variable |
| Y1 | First stage response variable |
| X2 | Second stage explanatory variable |
| Y2 | Second stage response variable |
| L2 | Ssecond stage lower bound of CI |
| U2 | Second stage upper bound of CI |
| call | Method Call |
| sigmaSq | Estimate of variance |
| deriv_d0 | Derivative estimate |
| class | twostageTE |

## Author(s)

Shawn Mankad

## Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
oneStage_IR=stageOneAnalysis(X, Y, 0.25, type="IR-wald", 0.99)
X2=runif(75,oneStage_IR$L1 ,oneStage_IR$U1)
Y2=X2^2+rnorm(n=length(X2), sd=0.1)
twoStage_IR = stageTwoAnalysis(oneStage_IR, X2, Y2, type="IR-wald", 0.95)

## The function is currently defined as
function (stageOne, explanatory, response, type = "IR-wald",
    level = 0.95, combineData = FALSE)
{
    cl1 <- match.call(expand.dots = TRUE)
    Y_0 = stageOne$threshold
    C_1 = stageOne$C_1
    gamma1=1/3
    if (combineData) {
explanatory = c(explanatory ,
    stageOne$X1[stageOne$X1 > stageOne$L1 & stageOne$X1 < stageOne$U1])
response = c(response ,
    stageOne$Y1[stageOne$X1 > stageOne$L1 & stageOne$X1 < stageOne$U1])
    }
    if (type == "IR-wald") {
        CI = waldConfidenceInterval_ir_stageTwo(explanatory,
```

```
            response, Y_0, level = level, gamma1 = gamma1, C_1 = C_1,
            n1 = length(stageOne$X1))
        return(structure(list(L2 = CI$lower, U2 = CI$upper, estimate = CI$estimate,
            threshold = Y_0, level = level, X1 = stageOne$X1,
            Y1 = stageOne$Y1, X2 = explanatory, Y2 = response,
            L1 = stageOne$L1, U1 = stageOne$U1, call = cl1, sigmaSq = CI$sigmaSq,
            deriv_d0 = CI$deriv_d0), class = "twostageTE"))
    }
    else if (type == "IR-likelihood") {
        CI = likelihoodConfidenceInterval(explanatory, response,
            Y_0, level = level)
        return(structure(list(L2 = CI$lower, U2 = CI$upper, estimate = CI$estimate,
            threshold = Y_0, level = level, X1 = stageOne$X1,
            Y1 = stageOne$Y1, X2 = explanatory, Y2 = response,
            L1 = stageOne$L1, U1 = stageOne$U1, call = cl1, sigmaSq = CI$sigmaSq,
            deriv_d0 = CI$deriv_d0), class = "twostageTE"))
    }
    else if (type == "SIR") {
        CI = waldConfidenceInterval_sir_stageTwo(explanatory = explanatory,
            response = response, Y_0 = Y_0, gamma1 = gamma1,
            C_1 = C_1, level = level)
        return(structure(list(L2 = CI$lower, U2 = CI$upper, estimate = CI$estimate,
            threshold = Y_0, level = level, X1 = stageOne$X1,
            Y1 = stageOne$Y1, X2 = explanatory, Y2 = response,
            L1 = stageOne$L1, U1 = stageOne$U1, call = cl1, sigmaSq = CI$sigmaSq,
            deriv_d0 = CI$deriv_d0), class = "twostageTE"))
    }
    else if (type == "locLinear") {
        CI = linearBootstrapConfidenceInterval_stageTwo(explanatory = explanatory,
            response = response, Y_0 = Y_0, level = level)
        return(structure(list(L2 = CI$lower, U2 = CI$upper, estimate = CI$estimate,
            threshold = Y_0, level = level, X1 = stageOne$X1,
            Y1 = stageOne$Y1, X2 = explanatory, Y2 = response,
            L1 = stageOne$L1, U1 = stageOne$U1, call = cl1, sigmaSq = CI$sigmaSq,
            deriv_d0 = CI$deriv_d0), class = "twostageTE"))
    }
    else error("stageOneAnalysis: type should be either
        'IR-wald','IR-likelihood', 'SIR', or 'locLinear'")
}
```

---

summary.twostageTE *summary method for object twostageTE*

---

## Description

summary method for object twostageTE

## Usage

```
## S3 method for class 'twostageTE'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | twostageTE object |
| ... | ignored |

### Value

prints confidence interval, point estimate, and auxiliary estimates

### Author(s)

Shawn Mankad

### Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
oneStage_IR=stageOneAnalysis(X, Y, 0.25, type="IR-wald", 0.99)
summary(oneStage_IR)
```

---

## threshold_estimate_ir *Threshold estimate based on IR*

---

### Description

Uses isotonic regression and PAVA to form a point estimate.

### Usage

```
threshold_estimate_ir(explanatory, response, Y_0)
```

### Arguments

| | |
|---|---|
| explanatory | Explanatory sample points |
| response | Observed responses at the explanatory sample points |
| Y_0 | Threshold of interest |

### Details

This is an internal function not meant to be called directly. It function relies on the PAVA algorithm to form a point estimate.

## Value

list(threshold_estimate_explanatory = estim_x, threshold_estimate_response = fit$y[ind], threshold = Y_0, Y_hat = fit$y, index = ind)

`threshold_estimate_explanatory`
: Point estimate of d_0

`threshold_estimate_response`
: Estimate of f(d_0), which may not be exactly equal to the desired threshold

`threshold`
: Threshold of interest (equal to Y_0 input)

`Y_hat`
: Fitted values from PAVA

`index`
: index that corresponds to the point estimate, so that Y_hat[index]=threshold_estimate_response

## Author(s)

Shawn Mankad

## Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
stageOneAnalysis(X, Y, 0.25, type="IR-wald", 0.99)

## The function is currently defined as
function (explanatory, response, Y_0)
{
    n = length(response)
    if (sum(response < Y_0) == n) {
        warning("Y_0 is outside observed region")
        list(threshold_estimate_explanatory = max(explanatory),
            threshold_estimate_response = max(response), threshold = Y_0,
            Y_hat = max(response), index = n)
    }
    else if (sum(response >= Y_0) == n) {
        warning("Y_0 is outside observed region")
        list(threshold_estimate_explanatory = min(explanatory),
            threshold_estimate_response = min(response), threshold = Y_0,
            Y_hat = min(response), index = 1)
    }
    else {
        fit = pava(explanatory, response)
        if (sum(fit$y >= Y_0) == 0) {
            warning("estimate is on the boundary")
            ind = n
            estim_x = fit$x[ind]
        }
        else if (sum(fit$y <= Y_0) == 0) {
            warning("estimate is on the boundary")
            ind = min(which(fit$y >= Y_0))
            estim_x = fit$x[ind]
        }
        else {
```

```
            ind = min(which(fit$y >= Y_0))
            estim_x = fit$x[ind]
        }
        list(threshold_estimate_explanatory = estim_x,
            threshold_estimate_response = fit$y[ind],
            threshold = Y_0, Y_hat = fit$y, index = ind)
    }
  }
```

---

`threshold_estimate_locLinear` *Threshold estimate based on local linear approximation*

---

## Description

The main idea for the procedure in Tang et al. (2011) is to utilize a local linear approximation in the vicinity of the first stage estimate, and to bootstrap this local approximation to obtain confidence intervals.

## Usage

```
threshold_estimate_locLinear(explanatory, response, Y_0)
```

## Arguments

| | |
|---|---|
| `explanatory` | Explanatory sample points |
| `response` | Observed responses at the explanatory sample points |
| `Y_0` | Threshold of interest |

## Details

This is an internal function not meant to be called directly. It function uses a local linear approximation to form a point estimate.

## Value

`threshold_estimate_explanatory`
: Point estimate of d_0

`threshold`
: Threshold of interest (equal to Y_0 input)

## Author(s)

Shawn Mankad

## References

Tang R, Banerjee M, Michailidis G (2011). 'A two-stage hybrid procedure for estimating an inverse regression function.' The Annals of Statistics, 39, 956-989.

## Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
oneStage_IR=stageOneAnalysis(X, Y, 0.25, type="IR-wald", 0.99)
X2 = c(rep(oneStage_IR$L1,37),rep(oneStage_IR$U1,38))
Y2=X2^2+rnorm(n=length(X2), sd=0.1)
stageTwoAnalysis(oneStage_IR, explanatory = X2, response = Y2,
type = "locLinear", level = 0.95)


## The function is currently defined as
function (explanatory, response, Y_0)
{
    n = length(response)
    if (sum(response < Y_0) == n) {
        list(threshold_estimate_explanatory = max(explanatory),
            threshold_estimate_response = max(response), threshold = Y_0,
            Y_hat = max(response), index = n)
    }
    else if (sum(response >= Y_0) == n) {
        list(threshold_estimate_explanatory = min(explanatory),
            threshold_estimate_response = min(response), threshold = Y_0,
            Y_hat = min(response), index = 1)
    }
    else {
        beta = lm(response ~ explanatory)$coef
        estim_x = (Y_0 - beta[1])/beta[2]
        list(threshold_estimate_explanatory = estim_x, threshold = Y_0)
    }
  }
```

---

`twostageTE` *Threshold value estimation using two-stage plans*

---

## Description

This package implements a variety of nonparametric methods for computing one-stage and two-stage confidence intervals and point estimates of threshold values.

## Details

| | |
|---|---|
| Package: | twostageTE |
| Type: | Package |
| Version: | 1.0 |
| Date: | 2013-05-23 |
| License: | GPL-2 |

The user interacts with the package by utilizing two functions: stageOneAnalysis and stageTwoAnalysis. These functions take the sampled explanatory variable and corresponding responses at the first and second stage, respectively, and outputs point estimate and confidence intervals based on different user specific procedures.

## Author(s)

Shawn Mankad Maintainer: Shawn Mankad <smankad@umich.edu>

## References

Shawn Mankad, George Michailidis, Moulinath Banerjee (2015). Threshold Value Estimation Using Adaptive Two-Stage Plans in R. Journal of Statistical Software, 67(3), 1-19. doi:10.18637/jss.v067.i03

## Examples

```
## Simulating the (wiggly) isotonic Sine function ##
sampleData=function(n, lower, upper) {
x=runif(n, lower, upper)
y=(1/40)*sin(6*pi*x) + 1/4 + x/2 + (1/4)*x^2
+ rnorm(n=length(x), sd=0.1)
return(list(X=x, Y=y))
}
Budget=100
d0=0.5
threshold = (1/40)*sin(6*pi*d0) + 1/4 + d0/2 + (1/4)*d0^2

n1=floor(Budget*0.25)
n2=Budget - n1
samp = sampleData(n1, lower=0, upper=1)
X = samp$X
Y = samp$Y
## Two Stage IR+IR ##
stageOne_IR=stageOneAnalysis(X, Y, threshold, type="IR-wald", 0.99)
samp2 = sampleData(n2, lower=stageOne_IR$L1, upper=stageOne_IR$U1)
X2 = samp2$X
Y2 = samp2$Y
twoStageIR = stageTwoAnalysis(stageOne_IR, X2, Y2, type="IR-wald", 0.95)
## Two Stage LR+LR ##
stageOne_LR=stageOneAnalysis(X, Y, threshold, type="IR-likelihood", 0.99)
samp2 = sampleData(n2, lower=stageOne_LR$L1, upper=stageOne_LR$U1)
X2 = samp2$X
Y2 = samp2$Y
twoStageLR = stageTwoAnalysis(stageOne_LR, X2, Y2,
    type="IR-likelihood", 0.95)
## Two Stage IR+Local Linear ##
X2 = c(rep(stageOne_IR$L1,37),rep(stageOne_IR$U1,38))
Y2=X2^2+rnorm(n=length(X2), sd=0.1)
twoStageLinear=stageTwoAnalysis(stageOne_IR, explanatory = X2, response = Y2,
    type = "locLinear", level = 0.95)
```

`waldConfidenceInterval_ir_stageOne` *Stage one IR-Wald confidence interval*

## Description

This is an internal function not meant to be called directly. Classical IR-Wald confidence interval that can be called at the first stage of a multistage procedure

## Usage

```
waldConfidenceInterval_ir_stageOne(explanatory, response, Y_0, level = NA)
```

## Arguments

| | |
|---|---|
| `explanatory` | Explanatory sample points |
| `response` | Observed responses at the explanatory sample points |
| `Y_0` | Threshold of interest |
| `level` | Desired confidence level |

## Value

| | |
|---|---|
| `estimate` | Point estimate for d_0 |
| `lower` | Lower bound of the confidence interval |
| `upper` | upper bound of the confidence interval |
| `C_1` | Constant for computing the confidence interval – required for second stage ir-wald analysis |
| `sigmaSq` | Estimate of variance |
| `deriv_d0` | Estimate of the derivative at d_0 |

## Author(s)

Shawn Mankad

## Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
oneStage_IR=stageOneAnalysis(X, Y, 0.25, type="IR-wald", 0.99)

## The function is currently defined as
function (explanatory, response, Y_0, level = NA)
{
    if (is.na(level)) {
        level = 0.95
    }
```

```
    alpha = 1 - level
    ## Import previously computed Chernoff quantiles, provided by Groeneboom and Wellner
    chernoff_realizations <- NULL; rm(chernoff_realizations);
    data("chernoff_realizations", envir =environment())
    ind = min(which(chernoff_realizations$DF - (1-alpha/2) >= 0))
    q = chernoff_realizations$xcoor[ind]
    n = length(response)

    fit = threshold_estimate_ir(explanatory, response, Y_0)
    sigmaSq = estimateSigmaSq(explanatory, response)$sigmaSq
    deriv_d0 = estimateDeriv(explanatory, response,
        fit$threshold_estimate_explanatory, sigmaSq)
    g_d0 = 1/n

    n = length(explanatory)
    C_di = (4 * sigmaSq/(deriv_d0^2))^(1/3)
    band = n^(-1/3) * C_di * g_d0^(-1/3) * q
    return(list(estimate = fit$threshold_estimate_explanatory,
        lower = max(min(explanatory), fit$threshold_estimate_explanatory -
            band), upper = min(max(explanatory), fit$threshold_estimate_explanatory +
            band), C_1 = as.numeric(C_di * g_d0^(-1/3) * q),
        sigmaSq = sigmaSq, deriv_d0 = deriv_d0))
}
```

---

`waldConfidenceInterval_ir_stageTwo` *Two-stage IR-Wald confidence interval*

---

## Description

This is an internal function not meant to be called directly. IR-Wald confidence interval that can be called at the second stage of a multistage procedure

## Usage

```
waldConfidenceInterval_ir_stageTwo(explanatory, response,
    Y_0, gamma1, C_1, n1, level = NA)
```

## Arguments

| | |
|---|---|
| `explanatory` | Explanatory sample points |
| `response` | Observed responses at the explanatory sample points |
| `Y_0` | Threshold of interest |
| `gamma1` | Constant that is used in the first stage confidence interval |
| `C_1` | Constant that is used in the first stage confidence interval |
| `n1` | Sample size of the first stage |
| `level` | Desired confidence level |

## Value

| | |
|---|---|
| estimate | Point estimate for d_0 |
| lower | Lower bound of the confidence interval |
| upper | upper bound of the confidence interval |
| sigmaSq | Estimate of variance |
| deriv_d0 | Estimate of the derivative at d_0 |

## Author(s)

Shawn Mankad

## References

Tang R, Banerjee M, Michailidis G, Mankad S (2013). 'Two-Stage Plans for Estimating a Threshold Value of a Regression Function.' http://arxiv.org/abs/1304.4637

## Examples

```
X=runif(25, 0,1)
Y=X^2+rnorm(n=length(X), sd=0.1)
oneStage_IR=stageOneAnalysis(X, Y, 0.25, type="IR-wald", 0.99)
X2=runif(75,oneStage_IR$L1 ,oneStage_IR$U1)
Y2=X2^2+rnorm(n=length(X2), sd=0.1)
twoStage_IR_IR = stageTwoAnalysis(oneStage_IR, X2, Y2, type="IR-wald", 0.95)

## The function is currently defined as
function (explanatory, response, Y_0, gamma1, C_1, n1, level = NA)
{
    if (is.na(level)) {
        level = 0.95
    }
    alpha = 1 - level
    chernoff_realizations <- NULL; rm(chernoff_realizations);
    data("chernoff_realizations", envir =environment())

    ind = min(which(chernoff_realizations$DF - (1-alpha/2) >= 0))
    q = chernoff_realizations$xcoor[ind]
    n = length(response)
    fit = threshold_estimate_ir(explanatory, response, Y_0)
    phi_0 = C_1 * n1 * (n^(-1))
    sigmaSq = estimateSigmaSq(explanatory, response)$sigmaSq
    deriv_d0 = estimateDeriv(explanatory, response, fit$threshold_estimate_explanatory,
        sigmaSq)
    C_di = (4 * sigmaSq/(deriv_d0^2))^(1/3)
    n = length(explanatory)
    p = gamma1/(1 + gamma1)
    C_di2 = C_di * (C_1/((1 - p) * p^(gamma1) * phi_0))
    band = n^(-1 * (1 + gamma1)/3) * C_di2 * q
    return(list(estimate = fit$threshold_estimate_explanatory,
        lower = max(min(explanatory), fit$threshold_estimate_explanatory -
```

```
            band), upper = min(max(explanatory), fit$threshold_estimate_explanatory +
            band), sigmaSq = sigmaSq, deriv_d0 = deriv_d0))
    }
```

# Index