Package ‘trade’

October 14, 2022

Type Package
Title Tools for Trade Practitioners
Version 0.8.1
Date 2022-08-23
Author Charles Taragin
Maintainer Charles Taragin <ctaragin+trader@gmail.com>
Depends antitrust (>= 0.99.11)
URL https://github.com/luciu5/trade
Imports methods, stats
Suggests BB,competitiontoolbox,rmarkdown,bookdown,knitr
VignetteBuilder knitr
Description A collection of tools for trade practitioners, including the ability to calibrate different consumer demand systems and simulate the effects of tariffs and quotas under different competitive regimes. These tools are derived from Anderson et al. (2001) <doi:10.1016/S0047-2727(00)00085-2> and Froeb et al. (2003) <doi:10.1016/S0304-4076(02)00166-5>.
License CC0
Encoding UTF-8
LazyLoad yes
RoxygenNote 7.2.0
Collate 'TariffClasses.R' 'ps-methods.R' 'HypoMonMethods.R'
  'QuotaClasses.R' 'summary-methods.R' 'bargaining_tariff.R'
  'SimFunctions.R' 'TariffCournot-methods.R'
  'TariffMonComRUM-methods.R' 'auction2nd_tariff.R'
  'bertrand_quota.R' 'bertrand_tariff.R' 'cournot_tariff.R'
  'initialize-methods.R' 'monopolistic_competition_tariff.R'
  'trade-deprecated.R' 'trade_shiny.R'
NeedsCompilation no
Repository CRAN
Date/Publication 2022-08-24 07:10:06 UTC
R topics documented:

- auction2nd_tariff .................................................. 2
- bargaining_tariff .................................................... 4
- bertrand_quota ....................................................... 7
- bertrand_tariff ..................................................... 9
- cournot_tariff ....................................................... 12
- defineMarketTools-methods ........................................... 15
- initialize-methods .................................................. 16
- monopolistic_competition_tariff ............................... 17
- ps-methods ........................................................... 19
- Quota-classes ....................................................... 19
- Sim-Functions ....................................................... 20
- summary-methods ................................................... 23
- Tariff-classes ....................................................... 24
- TariffCournot-methods ............................................. 24
- TariffMonComRUM-methods ......................................... 25

Index 27

---

auction2nd_tariff

Tariff Simulation With A Second Score Procurement Auction Game

Description

Simulate the effect of tariffs when firms play a second score procurement auction game and consumer demand is Logit.

Usage

```r
auction2nd_tariff(
  demand = c("logit"),
  prices,
  quantities,
  margins,
  owner = NULL,
  mktElast = NA_real_,
  diversions,
  tariffPre = rep(0, length(quantities)),
  tariffPost = rep(0, length(quantities)),
  priceStart,
  parmStart,
  control.slopes,
  control.equ,
  labels = paste("Prod", 1:length(quantities), sep = ""),
  ...)
```

**auction2nd_tariff**

**Arguments**

- **demand**: A character vector indicating which demand system to use. Currently allows logit (default).
- **prices**: A length k vector product prices.
- **quantities**: A length k vector of product quantities.
- **margins**: A length k vector of product margins. All margins must be in levels (not w.r.t to price), or NA.
- **owner**: EITHER a vector of length k whose values indicate which firm produced a product before the tariff OR a k x k matrix of pre-merger ownership shares.
- **mktElast**: A negative number equal to the industry pre-merger price elasticity. Default is NA.
- **diversions**: A k x k matrix of diversion ratios with diagonal elements equal to -1. Default is missing, in which case diversion according to revenue share is assumed.
- **tariffPre**: A vector of length k where each element equals the current ad valorem tariff (expressed as a proportion of the consumer price) imposed on each product. Default is 0, which assumes no tariff.
- **tariffPost**: A vector of length k where each element equals the new ad valorem tariff (expressed as a proportion of the consumer price) imposed on each product. Default is 0, which assumes no tariff.
- **priceStart**: For aids, a vector of length k who elements equal to an initial guess of the proportional change in price caused by the merger. The default is to draw k random elements from a [0,1] uniform distribution. For ces and logit, the default is prices.
- **parmStart aids only**: A vector of length 2 whose elements equal to an initial guess for each "known" element of the diagonal of the demand matrix and the market elasticity.
- **control.slopes**: A list of optim control parameters passed to the calibration routine optimizer (typically the calcSlopes method).
- **control.equ**: A list of BBsolve control parameters passed to the non-linear equation solver (typically the calcPrices method).
- **labels**: A k-length vector of labels.
- **...**: Additional options to feed to the BBsolve optimizer used to solve for equilibrium prices.

**Details**

Let k denote the number of products produced by all firms. Using price, and quantity, information for all products in each market, as well as margin information for at least one products in each market, auction2ndtariff is able to recover the slopes and intercepts of a Logit, CES, demand system. These parameters are then used to simulate the price effects of an ad valorem tariff under the assumption that the firms are playing a 2nd score auction.

**Value**

auction2ndtariff returns an instance of class Tariff2ndLogit
bargaining_tariff

References


See Also

bertrand_tariff to simulate the effects of a tariff under a Bertrand pricing game and monopolistic_competition_tariff to simulate the effects of a tariff under monopolistic competition.

Examples

```r
## Calibration and simulation results from a 10% tariff on non-US beers "OTHER-LITE" and "OTHER-REG"
## Source: Epstein/Rubenfeld 2004, pg 80

prodNames <- c("BUD","OLD STYLE","MILLER","MILLER-LITE","OTHER-LITE","OTHER-REG")
owner <-c("BUD","OLD STYLE","MILLER","MILLER","OTHER-LITE","OTHER-REG")
price <- c(.0441,.0328,.0409,.0396,.0387,.0497)
quantities <- c(.066,.172,.253,.187,.099,.223)*100
margins <- c(.3830,.5515,.5421,.5557,.4453,.3769) # margins in terms of price
margins <- margins*price # dollar margins
tariff <- c(0,0,0,0,.1,.1)

names(price) <- names(quantities) <- names(margins) <- prodNames

result.2nd <- auction2nd_tariff(demand = "logit",prices=price,quantities=quantities,
                                margins = margins,owner=owner,
                                tariffPost = tariff, labels=prodNames)

print(result.2nd) # return predicted price change
summary(result.2nd) # summarize merger simulation
```

bargainingTariff

Tariff Simulation With A Nash Bargaining Game

Description

Simulate the effect of tariffs when firms play a Nash Bargaining game and consumer demand is Logit.
Usage

```r
bargaining_tariff(
  demand = c("logit"),
  prices,
  shares,
  margins,
  owner = NULL,
  mktElast = NA_real_,
  insideSize = NA_real_,
  diversions,
  tariffPre = rep(0, length(shares)),
  tariffPost = rep(0, length(shares)),
  bargpowerPre = rep(0.5, length(prices)),
  bargpowerPost = bargpowerPre,
  normIndex = ifelse(isTRUE(all.equal(sum(shares), 1, check.names = FALSE)), 1, NA),
  priceOutside = ifelse(demand == "logit", 0, 1),
  priceStart,
  control.slopes,
  control.equ,
  labels = paste("Prod", 1:length(shares), sep = ""),
  ...
)
```

Arguments

- **demand**: A character vector indicating which demand system to use. Currently allows logit (default).
- **prices**: A length k vector product prices.
- **shares**: A length k vector of product shares. Values must be between 0 and 1.
- **margins**: A length k vector of product margins. All margins must be in levels (not w.r.t to price), or NA.
- **owner**: EITHER a vector of length k whose values indicate which firm produced a product before the tariff OR a k x k matrix of pre-merger ownership shares.
- **mktElast**: A negative number equal to the industry pre-merger price elasticity. Default is NA.
- **insideSize**: An integer equal to total pre-merger units sold. If shares sum to one, this also equals the size of the market.
- **diversions**: A k x k matrix of diversion ratios with diagonal elements equal to -1. Default is missing, in which case diversion according to revenue share is assumed.
- **tariffPre**: A vector of length k where each element equals the current ad valorem tariff (expressed as a proportion of the consumer price) imposed on each product. Default is 0, which assumes no tariff.
- **tariffPost**: A vector of length k where each element equals the new ad valorem tariff (expressed as a proportion of the consumer price) imposed on each product. Default is 0, which assumes no tariff.
bargpowerPre  A length k vector of pre-tariff bargaining power parameters. Values must be between 0 (sellers have the power) and 1 (buyers the power). NA values are allowed, though must be calibrated from additional margin and share data. Default is 0.5.

bargpowerPost A length k vector of post-tariff bargaining power parameters. Values must be between 0 (sellers have the power) and 1 (buyers the power). NA values are allowed, though must be calibrated from additional margin and share data. Default is ‘bargpowerPre’.

normIndex An integer equalling the index (position) of the inside product whose mean valuation will be normalized to 1. Default is 1, unless ‘shares’ sum to less than 1, in which case the default is NA and an outside good is assumed to exist.

priceOutside price of the outside good. Equals 0 for logit and 1 for ces. Not used for aids.

priceStart For aids, a vector of length k who elements equal to an initial guess of the proportional change in price caused by the merger. The default is to draw k random elements from a [0,1] uniform distribution. For ces and logit, the default is prices.

control.slopes A list of optim control parameters passed to the calibration routine optimizer (typically the calcSlopes method).

control.equ A list of BBsolve control parameters passed to the non-linear equation solver (typically the calcPrices method).

labels A k-length vector of labels.

... Additional options to feed to the BBsolve optimizer used to solve for equilibrium prices.

Details

Let k denote the number of products produced by all firms. Using price, and quantity, information for all products in each market, as well as margin information for at least one products in each market, bargaining_tariff is able to recover the slopes and intercepts of a Logit demand system. These parameters are then used to simulate the price effects of an ad valorem tariff under the assumption that the firms are playing a Nash Bargaining game.

Value

bargaining_tariff returns an instance of class TariffBargainingLogit

References


See Also

bertrand_tariff to simulate the effects of a tariff under a Bertrand pricing game and monopolistic_competition_tariff to simulate the effects of a tariff under monopolistic competition.
Examples

## Calibration and simulation results from a 10% tariff on non-US beers "OTHER-LITE" and "OTHER-REG"
## Source: Epstein/Rubenfeld 2004, pg 80

```r
prodNames <- c("BUD","OLD STYLE","MILLER","MILLER-LITE","OTHER-LITE","OTHER-REG")
owner <-c("BUD","OLD STYLE","MILLER","MILLER","OTHER-LITE","OTHER-REG")
price <= c(0.0441,0.0328,0.0409,0.0396,0.0387,0.0497)
shares <= c(0.066,0.172,0.253,0.187,0.099,0.223)
margins <= c(0.3830,0.5515,0.5421,0.557,0.4453,0.3769) # margins in terms of price
tariff <= c(0,0,0,0,.1,.1)

result.barg <- bargaining_tariff(demand = "logit",prices=price,shares=shares,
                                  margins = margins,owner=owner,
                                  tariffPost = tariff, labels=prodNames)

print(result.barg) # return predicted price change
summary(result.barg) # summarize merger simulation
```

bertrand_quota

### quota Simulation With A Bertrand Pricing Game

#### Description

Simulate the effect of quotas when firms play a Bertrand pricing game and consumer demand is either Logit, CES, or AIDS

#### Usage

```r
bertrand_quota(  
  demand = c("logit"),
  prices,
  quantities,
  margins,
  owner = NULL,
  mktElast = NA_real_,
  diversions,
  quotaPre = rep(Inf, length(quantities)),
  quotaPost,
  priceOutside = ifelse(demand == "logit", 0, 1),
```
Arguments

demand
A character vector indicating which demand system to use. Currently allows
logit (default), ces, or aids.

prices
A length k vector product prices. Default is missing, in which case demand
intercepts are not calibrated.

quantities
A length k vector of product quantities.

margins
A length k vector of product margins. All margins must be either be between 0
and 1, or NA.

owner
EITHER a vector of length k whose values indicate which firm produced a prod-
uct before the merger OR a k x k matrix of pre-merger ownership shares.

mktElast
A negative number equal to the industry pre-merger price elasticity. Default is
NA.

diversions
A k x k matrix of diversion ratios with diagonal elements equal to -1. Default is
missing, in which case diversion according to revenue share is assumed.

quotaPre
A vector of length k where each element equals the current quota (expressed as
a proportion of pre-merger quantities) imposed on each product. Default is Inf,
which assumes no quota.

quotaPost
A vector of length k where each element equals the new quota (expressed as a
proportion of pre-merger quantities) imposed on each product. Default is Inf,
which assumes no quota.

priceOutside
price of the outside good. Equals 0 for logit and 1 for ces. Not used for aids.

priceStart
For aids, a vector of length k who elements equal to an initial guess of the
proportional change in price caused by the merger. The default is to draw k
random elements from a [0,1] uniform distribution. For ces and logit, the default
is prices.

isMax
If TRUE, checks to see whether computed price equilibrium locally maximizes
firm profits and returns a warning if not. Default is FALSE.

parmStart
aids only. A vector of length 2 who elements equal to an initial guess for
"known" element of the diagonal of the demand matrix and the market elasticity.

control.slopes
A list of optim control parameters passed to the calibration routine optimizer
(typically the calcSlopes method).

control.equ
A list of BBsolve control parameters passed to the non-linear equation solver
(typically the calcPrices method).

labels
A k-length vector of labels.

... Additional options to feed to the BBsolve optimizer used to solve for equilib-
rium prices.
Details

Let \( k \) denote the number of products produced by all firms. Using price, and quantity, information for all products in each market, as well as margin information for at least one products in each market, \( \text{bertrand_quota} \) is able to recover the slopes and intercepts of the Logit, demand system. These parameters are then used to simulate the price effects of a quota under the assumption that the firms are playing a simultaneous price setting game.

Value

\( \text{bertrand_quota} \) returns an instance of class \( \text{QuotaLogit} \).

References


Examples

```r
## Calibration and simulation results from a 80% quota on non-US beers "OTHER-LITE" and "OTHER-REG"
## Source: Epstein/Rubenfeld 2004, pg 80
prodNames <- c("BUD","OLD STYLE","MILLER","MILLER-LITE","OTHER-LITE","OTHER-REG")
owner <-c("BUD","OLD STYLE","MILLER","MILLER","OTHER-LITE","OTHER-REG")
price <- c(.0441,.0328,.0409,.0396,.0387,.0497)
quantities <- c(.066,.172,.253,.187,.099,.223)*100
margins <- c(.3830,.5515,.5421,.5557,.4453,.3769)
quota <- c(Inf,Inf,Inf,Inf,.8,.8)
names(price) <- names(quantities) <- names(margins) <- prodNames

result.logit <- bertrand_quota(demand = "logit",prices=price,quantities=quantities,
margins = margins,owner=owner, quotaPost = quota, labels=prodNames)

print(result.logit) # return predicted price change
summary(result.logit) # summarize merger simulation
```

Tariff Simulation With A Bertrand Pricing Game

Description

Simulate the effect of tariffs when firms play a Bertrand pricing game and consumer demand is either Logit, CES, or AIDS.
Usage

bertrand_tariff(  
demand = c("logit", "ces", "aids"),  
prices,  
quantities,  
margins,  
owner = NULL,  
mktElast = NA_real_,  
diversions,  
tariffPre = rep(0, length(quantities)),  
tariffPost = rep(0, length(quantities)),  
priceOutside = ifelse(demand == "logit", 0, 1),  
priceStart,  
isMax = FALSE,  
parmStart,  
control.slopes,  
control.equ,  
labels = paste("Prod", 1:length(quantities), sep = ""),  
...  
)

Arguments

demand A character vector indicating which demand system to use. Currently allows logit (default), ces, or aids.

prices A length k vector product prices. Default is missing, in which case demand intercepts are not calibrated.

quantities A length k vector of product quantities.

margins A length k vector of product margins. All margins must be either between 0 and 1, or NA.

owner EITHER a vector of length k whose values indicate which firm produced a product before the tariff OR a k x k matrix of pre-merger ownership shares.

mktElast A negative number equal to the industry pre-merger price elasticity. Default is NA.

diversions A k x k matrix of diversion ratios with diagonal elements equal to -1. Default is missing, in which case diversion according to revenue share is assumed.

tariffPre A vector of length k where each element equals the current ad valorem tariff (expressed as a proportion of the consumer price) imposed on each product. Default is 0, which assumes no tariff.

tariffPost A vector of length k where each element equals the new ad valorem tariff (expressed as a proportion of the consumer price) imposed on each product. Default is 0, which assumes no tariff.

priceOutside price of the outside good. Equals 0 for logit and 1 for ces. Not used for aids.

priceStart For aids, a vector of length k who elements equal to an initial guess of the proportional change in price caused by the merger. The default is to draw k
random elements from a [0,1] uniform distribution. For ces and logit, the default is prices.

isMax
If TRUE, checks to see whether computed price equilibrium locally maximizes firm profits and returns a warning if not. Default is FALSE.

parmStart
aids only. A vector of length 2 whose elements equal to an initial guess for each “known” element of the diagonal of the demand matrix and the market elasticity.

control.slopes
A list of optim control parameters passed to the calibration routine optimizer (typically the calcSlopes method).

control.equ
A list of BBsolve control parameters passed to the non-linear equation solver (typically the calcPrices method).

labels
A k-length vector of labels.

... Additional options to feed to the BBsolve optimizer used to solve for equilibrium prices.

Details
Let k denote the number of products produced by all firms. Using price, and quantity, information for all products in each market, as well as margin information for at least one products in each market, bertrand_tariff is able to recover the slopes and intercepts of either a Logit, CES, or AIDS demand system. These parameters are then used to simulate the price effects of an ad valorem tariff under the assumption that the firms are playing a simultaneous price setting game.

Value
bertrand_tariff returns an instance of class TariffLogit, TariffCES, or TariffAIDS, depending upon the value of the “demand” argument.

References

See Also
monopolistic_competition_tariff to simulate the effects of a tariff under monopolistic competition.

Examples

```r
## Calibration and simulation results from a 10% tariff on non-US beers "OTHER-LITE" and "OTHER-REG"
## Source: Epstein/Rubenfeld 2004, pg 80

prodNames <- c("BUD","OLD STYLE","MILLER","MILLER-LITE","OTHER-LITE","OTHER-REG")
owner <-c("BUD","OLD STYLE","MILLER","MILLER","OTHER-LITE","OTHER-REG")
price <- c(.0441,.0328,.0409,.0396,.0387,.0497)
quantities <- c(.066,.172,.253,.187,.099,.223)*100
margins <- c(.3830,.5515,.5421,.5557,.4453,.3769)
```
cournot_tariff

Tariff Simulation With A Cournot Quantity Setting Game

Description

Simulate the effect of tariffs when firms play a cournot quantity setting game and consumer demand is either linear or log-linear

Usage

cournot_tariff(
  prices,
  quantities,
  margins = matrix(NA_real_, nrow(quantities), ncol(quantities)),
  demand = rep("linear", length(prices)),
  cost = rep("linear", nrow(quantities)),
  tariffPre = matrix(0, nrow = nrow(quantities), ncol = ncol(quantities)),
  tariffPost = tariffPre,
  mcfunPre = list(),
  mcfunPost = mcfunPre,
  vcfunPre = list(),
  vcfunPost = vcfunPre,
  capacitiesPre = rep(Inf, nrow(quantities)),
  capacitiesPost = capacitiesPre,
  productsPre = !is.na(quantities),
  productsPost = productsPre,
  owner = NULL,
  mktElast = rep(NA_real_, length(prices)),
  quantityStart = as.vector(quantities),
  control.slopes,
  control.equ,
  labels,
Arguments

prices A length k vector product prices.
quantities An n x k matrix of product quantities. All quantities must either be positive, or if the product is not produced by a plant, NA.
margins An n x k matrix of product margins. All margins must be either be between 0 and 1, or NA.
demand A length k character vector equal to "linear" if a product’s demand curve is assumed to be linear or "log" if a product’s demand curve is assumed to be log-linear.
cost A length k character vector equal to "linear" if a plant’s marginal cost curve is assumed to be linear or "constant" if a plant’s marginal curve is assumed to be constant. Returns an error if a multi-plant firm with constant marginal costs does not have capacity constraints.
tariffPre An n x k matrix where each element equals the current ad valorem tariff (expressed as a proportion of consumer price) imposed on each product. Default is 0, which assumes no tariff.
tariffPost An n x k matrix where each element equals the new ad valorem tariff (expressed as a proportion of consumer price) imposed on each product. Default is 0, which assumes no tariff.
mcfunPre a length n list of functions that calculate a plant’s marginal cost under the current tariff structure. If empty (the default), assumes quadratic costs.
mcfunPost a length n list of functions that calculate a plant’s marginal cost under the new tariff structure. If empty (the default), assumes quadratic costs.
vcfunPre a length n list of functions that calculate a plant’s variable cost under the current tariff structure. If empty (the default), assumes quadratic costs.
vfunPost a length n list of functions that calculate a plant’s variable cost under the new tariff structure. If empty (the default), assumes quadratic costs.
capacitiesPre A length n numeric vector of plant capacities under the current tariff regime. Default is Inf.
capacitiesPost A length n numeric vector of plant capacities under the new tariff regime. Default is Inf.
productsPre An n x k matrix that equals TRUE if under the current tariff regime, a plant produces a product. Default is TRUE if ‘quantities’ is not NA.
productsPost An n x k matrix that equals TRUE if under the new tariff regime, a plant produces a product. Default equals ‘productsPre’.
owner EITHER a vector of length n whose values indicate which plants are commonly owned OR an n x n matrix of ownership shares.
mktElast A length k vector of product elasticities. Default is a length k vector of NAs.
quantityStart A length k vector of quantities used as the initial guess in the nonlinear equation solver. Default is ‘quantities’.
control.slopes  A list of optim control parameters passed to the calibration routine optimizer (typically the calcSlopes method).

control.equ  A list of BBsolve control parameters passed to the non-linear equation solver (typically the calcPrices method).

labels  A k-length vector of labels.

...  Additional options to feed to the BBsolve optimizer used to solve for equilibrium quantities.

Details

Let k denote the number of products and n denote the number of plants. Using price, and quantity, information for all products in each market, as well as margin information for at least one products in each market, cournot.tariff is able to recover the slopes and intercepts of either a Linear or Log-linear demand system. These parameters are then used to simulate the price effects of a tariff under the assumption that the firms are playing a homogeneous products simultaneous quantity setting game.

Value

cournot.tariff returns an instance of class Cournot from package antitrust, depending upon the value of the "demand" argument.

References


Examples

```r
## Simulate the effect of a 75% ad valorem tariff in a 5-firm, single-product market with linear demand and quadratic costs
## Firm 1 is assumed to be foreign, and so subject to a tariff

n <- 5 # number of firms in market
cap <- rnorm(n, mean = .5, sd = .1)
int <- 10
slope <- -.25
tariffPre <- tariffPost <- rep(0, n)
tariffPost[1] <- .75

B.pre.c = matrix(slope, nrow=n, ncol=n)
diag(B.pre.c) = 2* diag(B.pre.c) - 1/cap
quantity.pre.c = rowSums(solve(B.pre.c) * -int)
price.pre.c = int + slope * sum(quantity.pre.c)
mc.pre.c = quantity.pre.c/cap
vc.pre.c = quantity.pre.c^2/(2*cap)
margin.pre.c = 1 - mc.pre.c/price.pre.c

# prep inputs for Cournot
```
owner.pre <- diag(n)

result.c <- cournot_tariff(prices = price.pre.c, quantities = as.matrix(quantity.pre.c),
     margins = as.matrix(margin.pre.c),
     owner = owner.pre,
     tariffPre = as.matrix(tariffPre),
     tariffPost = as.matrix(tariffPost))

summary(result.c, market = TRUE) # summarize tariff (high-level)
summary(result.c, market = FALSE) # summarize tariff (detailed)

Methods For Implementing The Hypothetical Monopolist Test

Description

An adaptation of the Hypothetical Monopolist Test described in the 2010 Horizontal Merger Guidelines for use in non-merger settings.

HypoMonTest implements the Hypothetical Monopolist Test for a given `ssnip`. ‘...’ may be used to pass arguments to the optimizer.

Usage

## S4 method for signature 'TariffBertrand'
HypoMonTest(object, prodIndex, ssnip = 0.05, ...)

## S4 method for signature 'TariffCournot'
HypoMonTest(object, plantIndex, prodIndex, ssnip = 0.05, ...)

Arguments

object An instance of one of the classes listed above.
prodIndex A vector of product indices that are to be placed under the control of the Hypothetical Monopolist.
ssnip A number between 0 and 1 that equals the threshold for a “Small but Significant and Non-transitory Increase in Price” (SSNIP). Default is .05, or 5%.
... Pass options to the optimizer used to solve for equilibrium prices.
plantIndex A vector of plant indices that are to be placed under the control of the Hypothetical Monopolist (Cournot).
HypoMonTest is an implementation of the Hypothetical Monopolist Test on the products indexed by ‘prodIndex’ for a ‘ssnip’. The Hypothetical Monopolist Test determines whether a profit-maximizing Hypothetical Monopolist who controls the products indexed by ‘prodIndex’ would increase the price of at least one of the products in ‘prodIndex’ by a small, significant, and non-transitory amount (i.e. impose a SSNIP). The main difference between this implementation and antitrust::HypoMonTest() is this implementation does not check to see if ‘prodIndex’ contains a merging party’s product.

HypoMonTest returns TRUE if a profit-maximizing Hypothetical Monopolist who controls the products indexed by ‘prodIndex’ would increase the price of at least one of the products in ‘prodIndex’ by a ‘ssnip’, and FALSE otherwise.

References


---

**Description**

Initialize methods for the TariffBertrand and TariffCournot classes

**Usage**

```r
## S4 method for signature 'TariffBertrand'
initialize(.Object, ...)

## S4 method for signature 'QuotaBertrand'
initialize(.Object, ...)

## S4 method for signature 'TariffCournot'
initialize(.Object, ...)
```

**Arguments**

- `.Object` an instance of class TariffBertrand or TariffCournot
- `...` arguments to pass to initialize
monopolistic_competition_tariff

**Tariff Simulation With A Monopolistic Competition Pricing Game**

**Description**

Simulate the effect of tariffs when firms play a Monopolistic Competition game and consumer demand is either Logit or CES

**Usage**

```r
monopolistic_competition_tariff(
  demand = c("logit", "ces"),
  prices,
  quantities,
  margins,
  mktElast = NA_real_,
  mktSize,
  tariffPre = rep(0, length(quantities)),
  tariffPost = rep(0, length(quantities)),
  priceOutside = ifelse(demand == "logit", 0, 1),
  labels = paste("Prod", 1:length(quantities), sep = "")
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>demand</td>
<td>A character vector indicating which demand system to use. Currently allows &quot;logit&quot; or &quot;ces&quot;.</td>
</tr>
<tr>
<td>prices</td>
<td>A length k vector product prices. Default is missing, in which case demand intercepts are not calibrated.</td>
</tr>
<tr>
<td>quantities</td>
<td>A length k vector of product quantities.</td>
</tr>
<tr>
<td>margins</td>
<td>A length k vector of product margins. All margins must be either be between 0 and 1, or NA.</td>
</tr>
<tr>
<td>mktElast</td>
<td>A negative number no greater than -1 equal to the industry pre-tariff price elasticity. Default is NA.</td>
</tr>
<tr>
<td>mktSize</td>
<td>A positive number equal to the industry pre-tariff market size. Market size equals total quantity sold, including sales to the outside good.</td>
</tr>
<tr>
<td>tariffPre</td>
<td>A vector of length k where each element equals the current ad valorem tariff (expressed as a proportion of the consumer price) imposed on each product. Default is 0, which assumes no tariff.</td>
</tr>
<tr>
<td>tariffPost</td>
<td>A vector of length k where each element equals the new ad valorem tariff (expressed as a proportion of the consumer price) imposed on each product. Default is 0, which assumes no tariff.</td>
</tr>
<tr>
<td>priceOutside</td>
<td>price of the outside good. Default 0 for logit and 1 for ces. Not used for aids.</td>
</tr>
<tr>
<td>labels</td>
<td>A k-length vector of labels.</td>
</tr>
</tbody>
</table>
**Details**

Let $k$ denote the number of products produced by all firms. Using price, and quantity, information for all products in each market, as well as margin information for at least one product in each market, `monopolistic_competition_tariff` is able to recover the slopes and intercepts of a Logit demand system. These parameters are then used to simulate the price effects of an *ad valorem* tariff under the assumption that the firms are playing a monopolistically competitive pricing game.

**Value**

`monopolistic_competition_tariff` returns an instance of class `TariffMonComLogit`, depending upon the value of the “demand” argument.

**References**


**See Also**

`bertrand_tariff` to simulate the effects of a tariff under a Bertrand pricing game.

**Examples**

```r
## Calibration and simulation results from a 10% tariff on non-US beers "OTHER-LITE" and "OTHER-REG"
## Source: Epstein/Rubenfeld 2004, pg 80

prodNames <- c("BUD","OLD STYLE","MILLER","MILLER-LITE","OTHER-LITE","OTHER-REG")
price <- c(.0441,.0328,.0409,.0396,.0387,.0497)
quantities <- c(.066,.172,.253,.187,.099,.223)*100
margins <- c(.3830,.5515,.5421,.5557,.4453,.3769)
tariff <- c(0,0,.1,0,.1,1)

names(price) <- names(quantities) <- names(margins) <- prodNames

result.logit <- monopolistic_competition_tariff(demand = "logit",prices=price,quantities=quantities,
margins = margins,
tariffPost = tariff, labels=prodNames)

print(result.logit) # return predicted price change
summary(result.logit) # summarize merger simulation

result.ces <- monopolistic_competition_tariff(demand = "ces",prices=price,quantities=quantities,
margins = margins,
tariffPost = tariff, labels=prodNames)
```

print(result.ces)  # return predicted price change
summary(result.ces) # summarize merger simulation

ps-methods

Methods To Calculate Producer Surplus

Description

Producer Surplus methods for the TariffBertrand and TariffCournot classes

Usage

## S4 method for signature 'TariffBertrand'
calcProducerSurplus(object, preMerger = TRUE)

## S4 method for signature 'TariffCournot'
calcProducerSurplus(object, preMerger = TRUE)

Arguments

object          an instance of class TariffBertrand or TariffCournot
preMerger       when TRUE, calculates producer surplus under the existing tariff regime. When FALSE, calculates tariffs under the new tariff regime. Default is TRUE.

Value

product-level (or in the case of Cournot, plant-level) producer surplus

Quota-classes

S4 classes to model quotas

Description

Extend classes from the antitrust package to accommodate quotas.

Slots

quotaPre For QuotaCournot, a matrix containing current plant-level (rows) AND product-level (columns) quotas. Default is a matrix of 0s. For all other classes, a vector containing current product-level quotas. Quotas are expressed as a proportion of pre-merger output. Default is a vector of Infs.

quotaPost a For QuotaCournot, a matrix containing new plant-level (rows) AND product-level (columns) quotas. Default is a matrix of Infs. For all other classes, a vector containing new product-level quotas. Quotas are expressed as a proportion of pre-merger output. Default is a vector of Infss.
Sim-Functions

Tariff Simulation With User-Supplied Demand Parameters

Description

Simulates the price effects of an ad valorem tariff with user-supplied demand parameters under the assumption that all firms in the market are playing either a differentiated products Bertrand pricing game, 2nd price auction, or bargaining game.

Let k denote the number of products produced by all firms below.

Usage

```r
sim(
  prices,
  supply = c("moncom", "bertrand", "auction", "bargaining"),
  demand = c("logit", "ces"),
  demand.param,
  owner,
  tariffPre = rep(0, length(prices)),
  tariffPost,
  subset = rep(TRUE, length(prices)),
  insideSize = 1,
  priceOutside,
  priceStart,
  bargpowerPre = rep(0.5, length(prices)),
  bargpowerPost = bargpowerPre,
  labels = paste("Prod", 1:length(prices), sep = ""),
  ...
)
```

Arguments

- **prices**: A length k vector of product prices.
- **supply**: A character string indicating how firms compete with one another. Valid values are "moncom" (monopolistic competition), "bertrand" (Nash Bertrand), "auction2nd" (2nd score auction), or "bargaining".
- **demand**: A character string indicating the type of demand system to be used in the merger simulation. Supported demand systems are logit ("Logit") or ces ("CES").
- **demand.param**: See Below.
- **owner**: EITHER a vector of length k whose values indicate which firm produced a product before the tariff OR a k x k matrix of pre-merger ownership shares.
- **tariffPre**: A vector of length k where each element equals the current ad valorem tariff (expressed as a proportion of the consumer price) imposed on each product. Default is 0, which assumes no tariff.
tariffPost  A vector of length k where each element equals the new ad valorem tariff (expressed as a proportion of the consumer price) imposed on each product. Default is 0, which assumes no tariff.

subset  A vector of length k where each element equals TRUE if the product indexed by that element should be included in the post-merger simulation and FALSE if it should be excluded. Default is a length k vector of TRUE.

insideSize  A length 1 vector equal to total units sold if ‘demand’ equals "logit", or total revenues if ‘demand’ equals "ces".

priceOutside  A length 1 vector indicating the price of the outside good. This option only applies to the 'Logit' class and its child classes. Default for ‘Logit’, ‘LogitNests’, and ‘LogitCap’ is 0, and for ‘CES’ and ‘CesNests’ is 1.

priceStart  A length k vector of starting values used to solve for equilibrium price. Default is the ‘prices’ vector for all values of demand except for ‘AIDS’, which is set equal to a vector of 0s.

bargpowerPre  A length k vector of pre-merger bargaining power parameters. Values must be between 0 (sellers have the power) and 1 (buyers the power). Ignored if ‘supply’ not equal to “bargaining”.

bargpowerPost  A length k vector of post-merger bargaining power parameters. Values must be between 0 (sellers have the power) and 1 (buyers the power). Default is ‘bargpowerPre’. Ignored if ‘supply’ not equal to “bargaining”.

labels  A k-length vector of labels. Default is “Prod#”, where ‘#’ is a number between 1 and the length of ‘prices’.

Details

Using user-supplied demand parameters, sim simulates the effects of a merger in a market where firms are playing a differentiated products pricing game.

If ‘demand’ equals ‘Logit’ then ‘demand.param’ must equal a list containing

• alpha The price coefficient.
• meanval A length-k vector of mean valuations ‘meanval’. If none of the values of ‘meanval’ are zero, an outside good is assumed to exist.

If demand equals ‘CES’ then ‘demand.param’ must equal a list containing

• gamma The price coefficient,
• alpha The coefficient on the numeraire good. May instead be calibrated using ‘shareInside’,
• meanval A length-k vector of mean valuations ‘meanval’. If none of the values of ‘meanval’ are zero, an outside good is assumed to exist,
• shareInside The budget share of all products in the market. Default is 1, meaning that all consumer wealth is spent on products in the market. May instead be specified using ‘alpha’.

Value

sim returns an instance of the class specified by the ‘demand’ argument.
Author(s)

Charles Taragin <ctaragin+trader@gmail.com>

See Also

The S4 class documentation for: Logit and CES.

Examples

```r
## Calibration and simulation results from a merger between Budweiser and
## Old Style. Note that the in the following model there is no outside
## good; BUD’s mean value has been normalized to zero.
## Source: Epstein/Rubenfeld 2004, pg 80

prodNames <- c("BUD","OLD STYLE","MILLER","MILLER-LITE","OTHER-LITE","OTHER-REG")
owner <-c("BUD","OLD STYLE","MILLER","MILLER","OTHER-LITE","OTHER-REG")
tariff <- c(0,0,0,.1,.1)
price <- c(.0441,.0328,.0409,.0396,.0387,.0497)

# a list containing price coefficient and mean valuations
demand.param=list(alpha=-48.0457,
                   meanval=c(0,0.4149233,1.1899885,0.8252482,0.1460183,1.4865730))

sim.logit <- sim(price,demand="logit",supply="bertrand", demand.param,
                     owner=owner,tariffPost=tariff,labels=prodNames)

print(sim.logit) # return predicted price change
summary(sim.logit) # summarize merger simulation
elast(sim.logit,TRUE) # returns premerger elasticities
elast(sim.logit,FALSE) # returns postmerger elasticities
diversion(sim.logit,TRUE) # return premerger diversion ratios
diversion(sim.logit,FALSE) # return postmerger diversion ratios

cmcr(sim.logit) #calculate compensating marginal cost reduction
upp(sim.logit) #calculate Upwards Pricing Pressure Index
CV(sim.logit) #calculate representative agent compensating variation
```
Summary methods for the `TariffBertrand`, `QuotaBertrand`, and `TariffCournot` classes

**Usage**

```r
## S4 method for signature 'TariffBertrand'
summary(
  object,
  revenue = FALSE,
  levels = FALSE,
  parameters = FALSE,
  market = FALSE,
  insideOnly = TRUE,
  digits = 2
)
```

```r
## S4 method for signature 'QuotaBertrand'
summary(
  object,
  revenue = FALSE,
  levels = FALSE,
  parameters = FALSE,
  market = FALSE,
  insideOnly = TRUE,
  digits = 2
)
```

```r
## S4 method for signature 'TariffCournot'
summary(
  object,
  market = FALSE,
  revenue = FALSE,
  levels = FALSE,
  parameters = FALSE,
  digits = 2
)
```

**Arguments**

- `object`: an instance of class `TariffBertrand`, `QuotaBertrand`, or `TariffCournot`
- `revenue`: When TRUE, returns revenues, when FALSE returns quantities. Default is FALSE.
levels When TRUE returns changes in levels rather than percents and quantities rather than shares, when FALSE, returns changes as a percent and shares rather than quantities. Default is FALSE.

parameters When TRUE, displays demand and cost parameters. Default is FALSE.

market When TRUE, displays aggregate information about the effect of a tariff. When FALSE displays product-specific (or in the case of Cournot, plant-specific) effects. Default is FALSE.

insideOnly When TRUE, rescales shares on inside goods to sum to 1. Default is FALSE.

digits Number of digits to report. Default is 2.

Value

Prints either market or product/plant-level summary and invisibly returns a data frame containing the same information.

Tariff-classes

S4 classes to model tariffs

Description

Extend classes from the antitrust package to accommodate tariffs.

Slots

tariffPre For TariffCournot, a matrix containing current plant-level (rows) AND product-level (columns) tariffs. Default is a matrix of 0s. For all other classes, a vector containing current product-level tariffs. ad valorem taxes are expressed as a proportion of the consumer price. Default is a vector of 0s.

tariffPost a For TariffCournot, a matrix containing new plant-level (rows) AND product-level (columns) tariffs. Default is a matrix of 0s. For all other classes, a vector containing new product-level tariffs. ad valorem taxes are expressed as a proportion of the consumer price. Default is a vector of 0s.

TariffCournot-methods Additional methods for TariffCournot Class

Description

Producer Surplus methods for the TariffBertrand and TariffCournot classes.
Usage

## S4 method for signature 'TariffCournot'
\texttt{calcSlopes(object)}

## S4 method for signature 'TariffCournot'
\texttt{calcQuantities(object, preMerger = TRUE, market = FALSE)}

Arguments

<table>
<thead>
<tr>
<th>object</th>
<th>an instance of class TariffCournot</th>
</tr>
</thead>
<tbody>
<tr>
<td>preMerger</td>
<td>when TRUE, computes result under the existing tariff regime. When FALSE, calculates tariffs under the new tariff regime. Default is TRUE.</td>
</tr>
<tr>
<td>market</td>
<td>when TRUE, computes market-wide results. When FALSE, calculates plant-specific results.</td>
</tr>
</tbody>
</table>

Value

calcSlopes return a TariffCournot object containing estimated slopes. CalcQuantities returns a matrix of equilibrium quantities under either the current or new tariff.

---

Description

calcSlopes, Prices, Margins methods for the TariffMonComLogit and TariffMonComCES classes

Usage

## S4 method for signature 'TariffMonComLogit'
\texttt{calcSlopes(object)}

## S4 method for signature 'TariffMonComCES'
\texttt{calcSlopes(object)}

## S4 method for signature 'TariffMonComLogit'
\texttt{calcMargins(object, preMerger = TRUE, level = FALSE)}

## S4 method for signature 'TariffMonComCES'
\texttt{calcMargins(object, preMerger = TRUE, level = FALSE)}

## S4 method for signature 'TariffMonComLogit'
\texttt{calcPrices(object, preMerger = TRUE, \ldots)}

## S4 method for signature 'TariffMonComCES'
\texttt{calcPrices(object, preMerger = TRUE, \ldots)}
TariffMonComRUM-methods

Arguments

object  an instance of class TariffMonComLogit or class TariffMonComCES
preMerger  when TRUE, computes result under the existing tariff regime. When FALSE, calculates tariffs under the new tariff regime. Default is TRUE.
level  when TRUE, computes margins in dollars. When FALSE, calculates margins as a proportion of prices. Default is FALSE.
...  harmlessly pass the arguments used in other calcPrices methods to methods for TariffMonComLogit and TariffMonComCES.

Value

calcSlopes return a TariffMonComLogit or TariffMonComCES object containing estimated slopes.
CalcQuantities returns a matrix of equilibrium quantities under either the current or new tariff.
Index

* methods
  defineMarketTools-methods, 15
antitrust::HypoMonTest(), 16
auction2nd_tariff, 2
bargaining_tariff, 4
BBsolve, 3, 6, 8, 11, 14
bertrand_quota, 7
bertrand_tariff, 4, 6, 9, 18
calcMargins,TariffMonComCES-method (TariffMonComRUM-methods), 25
calcMargins,TariffMonComLogit-method (TariffMonComRUM-methods), 25
calcPrices,TariffMonComCES-method (TariffMonComRUM-methods), 25
calcPrices,TariffMonComLogit-method (TariffMonComRUM-methods), 25
calcProducerSurplus,TariffBertrand-method (ps-methods), 19
calcProducerSurplus,TariffCournot-method (ps-methods), 19
calcQuantities,TariffCournot-method (TariffCournot-methods), 24
calcSlopes,TariffCournot-method (TariffCournot-methods), 24
calcSlopes,TariffMonComCES-method (TariffMonComRUM-methods), 25
calcSlopes,TariffMonComLogit-method (TariffMonComRUM-methods), 25
CES, 22
Cournot, 14
cournot_tariff, 12
defineMarketTools-methods, 15
HypoMonTest, 15
HypoMonTest (defineMarketTools-methods), 15
HypoMonTest,TariffBertrand-method (defineMarketTools-methods), 15
HypoMonTest,TariffCournot-method (defineMarketTools-methods), 15
initialize,QuotaBertrand-method (initialize-methods), 16
initialize,TariffBertrand-method (initialize-methods), 16
initialize,TariffCournot-method (initialize-methods), 16
initialize-methods, 16
Logit, 22
monopolistic_competition_tariff, 4, 6, 11, 17
optim, 3, 6, 8, 11, 14
ps-methods, 19
Quota-classes, 19
QuotaBertrand-class (Quota-classes), 19
QuotaCournot-class (Quota-classes), 19
QuotaLogit, 9
QuotaLogit-class (Quota-classes), 19
sim (Sim-Functions), 20
Sim-Functions, 20
summary,QuotaBertrand-method (summary-methods), 23
summary,TariffBertrand-method (summary-methods), 23
summary,TariffCournot-method (summary-methods), 23
summary-methods, 23
Tariff-classes, 24
Tariff2ndLogit, 3
Tariff2ndLogit-class (Tariff-classes), 24
TariffAIDS, 11
TariffAIDS-class (Tariff-classes), 24
TariffBargainingLogit, 6
TariffBargainingLogit-class (Tariff-classes), 24
TariffBertrand-class (Tariff-classes), 24
TariffCES. 11
TariffCES-class (Tariff-classes), 24
TariffCournot-class (Tariff-classes), 24
TariffCournot-methods, 24
TariffLogit, 11
TariffLogit-class (Tariff-classes), 24
TariffMonComLogit, 18
TariffMonComLogit-class (Tariff-classes), 24
TariffMonComRUM-methods, 25