The `antitrust` **Package**

Charles Taragin[*]   Michael Sandfort[*]

June 18, 2019

Contents

`antitrust` is a suite of tools that may be used in assessing the implications of horizontal mergers. The package contains functions that can calibrate the underlying parameters of a number of different supply and demand models as well as simulate the effects of a horizontal merger in different strategic environments. The output generated by these tools includes interesting features such as predicted price increases, welfare measures, demand elasticities, and the Hypothetical Monopolist Test. `antitrust` also includes functions that can assess the effects of a horizontal merger in other ways, including: *compensating marginal cost reduction* and *upwards pricing pressure.*

There are four features of `antitrust` that make it particularly useful for antitrust practitioners. First, `antitrust` collects a number of useful models onto a common platform, making it easy for practitioners to compare and contrast the results from different models.

Second, `antitrust` is open source software that runs on the `R` open source platform. Practically speaking, this means that practitioners not only have the flexibility to run this software wherever and whenever they wish, but they can also modify and extend the software as they see fit. We hope that having this collection of tools on a common, open source platform will facilitate discussion and collaboration among practitioners.

Third, `antitrust` includes a web interface built using `shiny`. While this interface does not give users acess to the full array of functionality in `antitrust`, it is simple to use and may provide first-time users, particularly those who are unfamiliar with `R`, a gentler introduction to `antitrust`. The interface may be invoked using the `antitrust_shiny` function.

Finally, the functions included in `antitrust` vary in the amount of information they require. Some functions, such as `upp.bertrand`, `cmcr.bertrand` and `cmcr.cournot` require only information on the merging parties' products, while functions like `linear`, `pcaids`, and `logit` require at least some information on all market participants. Table 7.1 summarizes the information requirements of all the functions included in `antitrust`.

The limited information needed for the economic models used in `antitrust` comes at some cost. First, the output of these models is sensitive to the supplied inputs. For instance, when employing the Bertrand model, inaccurate margins, shares and prices can yield inaccurate estimates of demand and cost parameters which can in turn yield incorrect predictions of a merger's effects. Calibrating model parameters with an array of plausible inputs will yield a range of outputs and illustrate the sensitivity of each model to those inputs.

Second, none of the parameters calibrated by `antitrust` may be used in frequentist statistical hypothesis testing. In other words, while the economic models in `antitrust` may be used to generate reliable estimates of the effects of the merger, statistical tests cannot be used to determine the accuracy of these estimates. Accomplishing this requires additional data and is beyond the current scope of `antitrust`.

This document provides an introduction to the economic theory upon which the `antitrust` packages' functions are built. Please use the `help` function for assistance invoking any of the functions, classes, or methods included in `antitrust`. In particular, note that the help pages for all the functions listed in Table 7.1 contain examples illustrating how to use the function.

Part I

Unilateral Effects

1 The Bertrand Pricing Game

Much of `antitrust`'s functionality is built around the Bertrand pricing game. This version of the game assumes that firms producing multiple differentiated products with distinct, constant, marginal costs simultaneously set their products' prices to maximize their profits. In this model, prices are *strategic complements* in the sense that increasing the price of one product causes some customers to switch to other products, raising the quantities sold and therefore the profit-maximizing prices of these other products. Ultimately, it is the magnitude of these lost sales that, at the margin, dissuades firms from raising their prices further .

Mergers are modeled by assuming that the merging parties' products are placed under common ownership, which, if the products are substitutes, allows the merged entity to *recapture* some of the sales that would otherwise be lost. As a result, the Bertrand model (for some demand systems) predicts that absent any efficiencies affecting the marginal cost of production, the prices of all of the merging parties' products will increase, and the price of all other products in the market will not decrease.

Currently, this version of the Bertrand model does not allow firms to add or reposition products, or allow firms to engage in some forms of price discrimination.[1]

1.1 The Game

Suppose that there are K firms in a market, and that each of the $k \in K$ firms produces n_k products.[2] Let $n = \sum_{k \in K} n_k$ denote the number of products sold by all K firms. The Bertrand model assumes that firms simultaneously set their products' prices in order to maximize their profits. This model also assumes that all firms can perfectly observe each others' prices, quantities, costs, and product characteristics.

Functions in `antitrust`'s Bertrand model also adopt the additional assumption that

---

[1]In particular, this version of the Bertrand model does not accommodate non-linear pricing, such as is used in 2nd or 3rd degree price discrimination.

[2]Throughout, we abuse the notation slightly by treating variables like K as both the set of firms as well as the number of firms.

each product is produced using its own distinct constant marginal cost technology c_i, for all $i \in n$. As we will see, this assumption is necessary when information is limited.

1.1.1 The Mathematical Model

Firm $k \in K$ chooses the prices $\{p_i\}_{i=1}^{n_k}$ of its products so as to maximize profits. Mathematically, firm k solves:

$$\max_{\{p_i\}_{i=1}^{n_k}} \sum_{i=1}^{n} \omega_{ik}(p_i - c_i)q_i,$$

where ω_{ik} is the share of product i's profits earned by firm k, so that $\sum_{k \in K} \omega_{ik} \leq 1$. q_i, the quantity sold of product i, is assumed to be a twice differentiable function of *all* product prices.

Differentiating profits with respect to each p_i yields the following first order conditions (FOCs):

$$\partial p_i \equiv \omega_{ik}q_i + \sum_{j=1}^{n} \omega_{jk}(p_j - c_j)\frac{\partial q_j}{\partial p_i} = 0 \qquad \text{for all } i \in n_k$$

which may be rewritten as

$$\partial p_i \equiv \omega_{ik}r_i + \sum_{j=1}^{n} \omega_{jk}r_j m_j \epsilon_{ji} = 0 \qquad \text{for all } i \in n_k,$$

where $r_i \equiv \frac{p_i q_i}{\sum_{j=1}^{n} p_j q_j}$ is product i's revenue share, $m_i \equiv \frac{p_i - c_i}{p_i}$ is product i's gross margin, and $\epsilon_{ij} \equiv \frac{\partial q_i}{\partial p_j}\frac{p_j}{q_i}$ is the elasticity of product i with respect to the price of product j.

The FOCs for all products may be stacked and then represented using the following matrix notation:

$$(r \circ diag(\Omega)) + (E \circ \Omega)'(r \circ m) = 0 \qquad (1.1.1)$$

rearranging yields

$$m_{Bertrand} = -\{(E \circ \Omega)'^{-1}(r \circ diag(\Omega))\} \circ \frac{1}{r} \qquad (1.1.2)$$

where r and m are n-length vectors of revenue shares and margins, $E = \begin{pmatrix} \epsilon_{11} & \cdots & \epsilon_{1n} \\ \vdots & \ddots & \vdots \\ \epsilon_{n1} & \cdots & \epsilon_{nn} \end{pmatrix}$
is a $n \times n$ matrix of own- and cross-price elasticities, and $\Omega = \begin{pmatrix} \omega_{11} & \cdots & \omega_{1n} \\ \vdots & \ddots & \vdots \\ \omega_{n1} & \cdots & \omega_{nn} \end{pmatrix}$ is an
$n \times n$ matrix whose i, jth element equals the share of product j's profits owned by the firm setting product i's price[3]. In many cases, product i and j are wholly owned by a single firm, in which cases the i, jth element of Ω equals 1 if i and j are owned by the same firm and 0 otherwise. Under partial ownership, the columns of the matrix formed from the unique rows of Ω must sum to 1. '$diag$' returns the diagonal of a square matrix and '$\circ$' is the Hadamard (entry-wise) product operator.

The solution to system 1.1.1 yields equilibrium prices conditional on the ownership structure Ω. A (partial) merger is modeled as the solution to system 1.1.1 where Ω is changed to reflect the change in ownership.

1.1.2 Adding Exogenous Capacity Constraints

The Bertrand model described above assumes that products are produced with constant marginal costs and no capacity constraints. Here, we extend this model to allow for exogenous capacity constraints.[4]

Firm $k \in K$ chooses the prices $\{p_i\}_{i=1}^{n_k}$ of its products so as to maximize profits, subject to capacity constraints $\{t_i\}_{i=1}^{n_k}$. Mathematically, firm k solves:

$$\max_{\{p_i\}_{i=1}^{n_k}} \sum_{i=1}^{n} \omega_{ik}(p_i - c_i)q_i,$$

subject to

$$q_i \leq t_i, \qquad\qquad\qquad i = 1 \ldots n_k$$

---

[3]The Bertrand model assumes that while any firm can receive a portion of another firm's profits (e.g. through owning a share of that firms' assets), only one firm can set a product's price.

[4]This section is based on the model described in ?, p. 51-55

9

In general, either the capacity constraint for product i will bind and the firm will be forced to produce less of i than it would find optimal, or the capacity constraint will not bind, and the firm will produce the optimal amount implied by the FOCs. In the former, it can be shown that $\partial p_i \leq 0$ and $q_i - t_i = 0$, while in the latter $\partial p_i = 0$ and $q_i - t_i \leq 0$. Mathematically, these cases can be written as

$$\max\{\partial p_i, q_i - t_i\} = 0, i = 1 \ldots n_k \qquad (1.1.3)$$

1.2 Calibrating Model Demand and Cost Parameters

Although the functions listed in the "Bertrand" section of Table 7.1 are based on the Bertrand model and use similar inputs, they can yield very different equilibrium price predictions. This can occur for two reasons. First, these functions use different demand systems with very different curvatures to simulate the price effects from a merger. Indeed, equation 1.1.1 indicates that it is these curvatures, embodied in the matrix of own- and cross-price elasticities E, that play an important role in calculating price effects.

Second, binding capacity constraints can limit the incentive of the merging parties to raise prices, or the ability of other firms in the market to respond to a price increase. If, pre-merger, none of the merging parties' products are capacity constrained but some of the other firms' products are, then post-merger equilibrium prices will typically be *higher* than if none of the capacity constraints were binding pre-merger. Also, if pre-merger, some of the merging parties' products are capacity constrained but none of the other firms' products are constrained, then post-merger equilibrium prices will typically be *lower* than if none of the capacity constraints were binding pre-merger.

For all the demand specifications listed in the "Bertrand" section of Table 7.1, the calibration strategy is the same. First, we assume that quantities/shares and (with the exception of LA-AIDS) prices are observed for *all* products in the market, and that margins for *some* products are observed. Our decision to treat quantities, prices, and margins as primitives comes directly from equation 1.1.1. For capacity-constrained models, equation 1.1.3 indicates that all product capacities must be observed as well.

In addition to quantities, prices, some margins and capacities, we assume that users observe diversion ratios. Diversion ratios come in two forms: *quantity* diversion and *revenue* diversion. The *quantity* diversion from product i to product j d_{ij}^q is defined as the percentage of all of i's lost unit sales that switch to j *due to a price increase*

in product i, while the *revenue* diversion from product i to product j d_{ij}^r is defined as the percentage of all of i's lost revenue that switches to j *due to a price increase in product i*. Mathematically, quantity and revenue diversion may be represented as

$$d_{ij}^q = -\frac{\frac{\partial q_j}{\partial p_i}}{\frac{\partial q_i}{\partial p_i}}$$

$$= -\frac{\epsilon_{ji} q_j}{\epsilon_{ii} q_i} \tag{1.2.1}$$

$$d_{ij}^r = -\frac{\frac{\partial p_j q_j}{\partial p_i}}{\frac{\partial p_i q_i}{\partial p_i}}$$

$$= -\frac{\epsilon_{ji}(\epsilon_{jj} - 1)r_j}{\epsilon_{jj}(\epsilon_{ii} - 1)r_i} \tag{1.2.2}$$

Note that d_{ij}^q, d_{ij}^r are restricted to be between -1 and 1, and are positive if products i and j are substitutes and negative if they are complements. Additionally, conditional on customers switching from product i, they must switch to another product (i.e. $\sum_j d_{ij} \leq 0$). For all the models included in `antitrust`, we assume that i and j are not complements ($d_{ij} \geq 0$) and for some demand models (i.e. AIDS) we will assume that $\sum_j d_{ij} = 0$.

Although diversion ratios are not present in either equation 1.1.1 or 1.1.3, these definitions indicate that diversion ratios may be helpful in recovering the matrix of own- and cross-price elasticities E. Indeed, for a number of the demand systems described below, diversions will be used for just this purpose.

We further assume that all of this information represents the outcome of the unique pre-merger equilibrium for firms in the market playing the static Bertrand pricing game described above. We then substitute observed margins, shares and prices into either equation 1.1.1 or 1.1.3, which is now solely a function of demand parameters, and then solve for the coefficient(s) on prices. Once the price coefficients have been estimated, we use observed prices and the demand equations to estimate the intercepts.

Often, there are more FOCs than unknown price coefficients. For instance, under Logit demand, there is only one price parameter that needs to be estimated and up to n FOCs with which to estimate it. This means that at a minimum, users need only supply enough margin information to complete a single product's FOC. If that product happens to be owned by a single-product firm, then only one margin is necessary. On the other hand, if the product happens to be owned by a multi-

product firm, then at a minimum, all the margins for products owned by that firm must be supplied.

The (Marshallian) demand specifications used in `antitrust` can be grouped into two categories: demand systems that are derived from a representative consumer's expenditure function and demand systems that are derived from a representative consumer's indirect utility function. The linear, log-linear, and LA-AIDS demand systems fall into the former category, while the Logit and CES fall into the latter category. Below, we briefly discuss these demand systems as well as the assumptions and/or data needed to recover estimates of the demand parameters.

We conclude this section with a discussion of how calibrated demand parameters and the FOCs can be used to calibrate product-specific constant marginal costs.

1.2.1 Linear Demand

The Bertrand model with linear demand may be implemented using the `linear` function.

The linear demand system assumes that the demand for each product $i \in n$ in the market is given by

$$q_i = \alpha_i + \sum_{j \in n} \beta_{ij} p_j \text{ for all } i \in n,$$

which may be written in matrix notation as

$$q = \alpha + Bp,$$

where $q \geq 0, p > 0$ are vectors of product quantities and prices, α is a vector of product specific demand intercepts and B is a matrix of slopes. This demand system yields the following own- and cross-price elasticities:

$$\epsilon_{ii} = \beta_{ii} \frac{p_i}{q_i}, \qquad\qquad \epsilon_{ii} < 0$$
$$\epsilon_{ij} = \beta_{ij} \frac{p_i}{q_j}, \qquad\qquad \epsilon_{ij} \geq 0$$

Without additional restrictions and/or data, there are $2n$ equations (n FOCs and n demand equations) but $n(n + 1)$ unknown parameters, which means that there

are more unknowns than equations and the demand parameters α, B cannot be recovered. To remedy this, we assume that the *quantity* diversion is observed.[5] With the linear model, this assumption increases the number of equations to $n(n+1)$, allowing estimates of α and B to be recovered if prices, quantities and margins are observed for all products.

One known issue with the linear demand system is that, while analytically tractable, it is not rooted in consumer choice theory. Indeed, it has been shown that the linear demand system without income effects is consistent with the axioms of consumer choice if B is a symmetric matrix and satisfies homogeneity of degree 0 in prices.[6] Imposing these additional assumptions reduces the number of unknown parameters to $\frac{n(n+3)}{2}$ ($\frac{n(n+1)}{2}$ elements of B and n intercepts), which means that the system is over-identified.[7]

The linear demand system can be modified to allow for substitution to an outside (numeraire) good. To accomplish this, assume that the price of the outside product is not set strategically (i.e. not set as part of the Nash-Bertrand game played by the manufacturers of the n products included in the simulation). It can be shown that if B is symmetric and satisfies homogeneity of degree 0 in prices when the outside good is included, then $\sum_{j \in n} d_{ij} < 0$.[8] In other words, under symmetry and homogeneity of degree 0, users may include a non-strategically priced outside good

---

[5] By default, `linear` assumes diversion according to quantity share. Diversion according to quantity share assumes that $d_{ij}^q = \frac{s_j}{1-s_i}$, where s_i, s_j are the *quantity* share of i and j. As we will see, this is the assumption underlying the Logit demand system.

[6] See [**?**, pp. 283].

[7] In fact, it turns out that only one element of B must be estimated. To see why, note that under symmetry, $\beta_{jj} = \frac{d_{ij}}{d_{ji}}\beta_{ii}$. Hence, if β_{ii} is known, then the preceding equation indicates that all the β_{jj}s may be recovered. From here, the definition of diversion may be used to recover all the β_{ij}s.

[8] To see this, note:

$$\sum_{i \in n} \beta_{ij} < 0 \qquad \text{(homogeneity of degree 0 with outside good)}$$

$$\Leftrightarrow \beta_{ii} + \sum_{j \neq i} \beta_{ji} < 0 \qquad \text{(symmetry)}$$

$$\Leftrightarrow \beta_{ii} - \sum_{j \neq i} d_{ij}\beta_{ii} < 0$$

$$\Leftrightarrow \beta_{ii}(1 - \sum_{j \neq i} d_{ij}) < 0$$

$$\Rightarrow \sum_{j \in n} d_{ij} < 0$$

where the last line follows since $\beta_{ii} < 0 \Rightarrow 1 - \sum_{j \neq i} d_{ij} > 0$.

by simply allowing the rows of the diversion matrix to sum to less than zero. The extent to which consumers substitute from product i to the outside good may be controlled by increasing the magnitude of $\sum_{j\in n} d_{ij}$.

By default, the `calcSlopes` method, called by the `linear` function to calibrate the linear demand parameters, assumes that the above assumptions hold and uses a minimum distance algorithm to find the elements of B that best satisfy i) all the FOCs for which there is sufficient information and ii) the diversion equations $d_{ij} = -\frac{\beta_{ji}}{\beta_{ii}}$, subject to the constraints that $\beta_{ij} \geq 0$ for all $i \neq j$ and $\sum_j b_{ij} \leq 0$ for all i. The model intercepts can then be recovered from the linear demand equations.

For completeness, `linear` includes the 'symmetry' argument that, when set equal to FALSE, instructs `calcSlopes` to calibrate demand parameters without imposing symmetry and homogeneity of degree zero in prices on B. Note that when 'symmetry' is FALSE, the system of equations is just-identified, which means that prices, quantities, and margins must be observed for all products. Also, note that when 'symmetry' is FALSE, Linear demand is unlikely to be consistent with consumer choice theory, and welfare measures such as compensating variation cannot be calculated.[9]

1.2.2 Log-Linear Demand

The Bertrand model with log-linear demand may be implemented using the `loglinear` function.

The log-linear demand system assumes that the demand for each product $i \in n$ in the market is given by

$$\log(q_i) = \alpha_i + \sum_{j\in n} \beta_{ij}\log(p_j) \text{ for all } i \in n, \qquad \beta_{ii} < 0$$

which may be written in matrix notation as

$$\log(q) = \alpha + B\log(p),$$

where q, p are vectors of product quantities and prices, α is a vector of product specific demand intercepts and B is a matrix of slopes. This demand system yields

---

[9]The `CV` method used to compute compensating variation checks to see if B is symmetric and returns an error if it isn't.

the following own- and cross-price elasticities:

$$\epsilon_{ii} = \beta_{ii}$$
$$\epsilon_{ij} = \beta_{ij}$$

As with linear demand, there are $2n$ equations but $n(n+1)$ unknown parameters, which means the demand parameters α, B cannot be recovered without additional assumptions. As before, we will assume that quantity diversion is known and by default occurs according to quantity share. However, it turns out that the parameter restrictions needed to make log-linear demand consistent with consumer choice theory are likely to be inconsistent with the Bertrand model.[10] As such, `loglinear` employs only the first assumption. Consequently, the demand parameters are just-identified, which means that users must supply `loglinear` with prices, margins, and quantities for all products in the market.

1.2.3 LA-AIDS Demand

The Bertrand model with the linear approximate Almost Ideal Demand System (LA-AIDS) may be implemented using the `aids` function.

The LA-AIDS without income effects assumes that the demand for each product $i \in n$ in the market is given by

$$r_i = \alpha_i + \sum_{j \in n} \beta_{ij} \log(p_j) \text{ for all } i \in n, \qquad \beta_{ii} < 0$$

which may be written in matrix notation as

$$r = \alpha + B \log(p),$$

---

[10]In order for log-linear demand without income effects to be consistent with consumer choice theory, either i) $\beta_{ij} = 1 + \beta_{ii}, -1 \neq \beta_{ii} \leq 0$ or ii) $\beta_{ij} = 0, \beta_{ii} = -1$ for all $i, j \in n$. Condition i) is unlikely to be true, since when products i and j are substitutes (typically the case we are most interested in evaluating), $\beta_{ij} > 0$ which in turn implies that $\beta_{ii} > -1$. However, if the owner of product i only manufacturers a single product (a typical occurrence), then the FOCs from the Bertrand model imply that $\beta_{ii} \leq -1$, a contradiction. Condition ii) is unlikely to hold since it implies that product i has no close substitutes and has marginal costs equal to 0. See **?** and **?** for more details.

where r, p are vectors of product *revenue* shares and prices, α is a vector of product-specific demand intercepts and B is a matrix of slopes[11].

The LA-AIDS model yields the following own- and cross-price elasticities:

$$\epsilon_{ii} = -1 + \frac{\beta_{ii}}{r_i} + r_i(1+\epsilon), \qquad\qquad \epsilon_{ii} < 0$$

$$\epsilon_{ij} = \frac{\beta_{ij}}{r_i} + r_j(1+\epsilon), \qquad\qquad \epsilon_{ij} \geq 0$$

where ϵ is the market elasticity of demand.

As with the linear demand system, the LA-AIDS model assumes that B is symmetric, satisfies homogeneity of degree zero in prices, and that diversion is known. The LA-AIDS model, however, assumes that *revenue* diversion, rather than *quantity* diversion is observed.[12] Under these two assumptions, there are $\frac{n(n+3)}{2}$ unknown demand parameters ($\frac{n(n-1)}{2}$ diagonal elements in B, n diagonal elements, and n intercepts) and up to $n(n+1)$ equations ($n(n-1)$ diversion equations, n FOCs and n demand equations), in which case the system is over-identified.[13]

One interesting feature of the LA-AIDS that distinguishes it from the linear and log-linear demand systems included in `antitrust` is that LA-AIDS elasticities incorporate ϵ, the market elasticity. Roughly speaking, ϵ controls the extent to which consumers substitute to products outside the n products included in the simulation given a small change in market-wide product prices. Here, we assume that ϵ is a parameter whose value is not a function of product prices.[14] While in some cases ϵ can be readily observed, in others it cannot. For the latter, the `calcSlopes` method (called by `aids`) exploits the fact that there are more equations than unknowns to identify both the unknown demand parameters described above as well as ϵ.

The `calcSlopes` method, called by the `aids` function to calibrate the AIDS parameters uses a minimum distance algorithm to find the ϵ and a single diagonal element of B that best satisfy i) all the FOCs for which there is sufficient information and ii) the diversion equations $d_{ij} = -\frac{\beta_{ji}}{\beta_{ii}}$, and iii) the market elasticity (if supplied using

---

[11]LA-AIDS differs from AIDS in that LA-AIDS substitutes the AIDS price index with Stone's price index. Since this version of LA-AIDS is without income effects, Stone's price index is only used to derive the own- and cross-price elasticities.

[12]If the 'diversion' argument to `aids` is missing, `aids` assumes diversion according to revenue share.

[13]In fact, it turns out that only one element of B must be estimated. To see why, note that under symmetry, $\beta_{jj} = \frac{d_{ij}}{d_{ji}}\beta_{ii}$. Hence, if β_{ii} is known, then the preceding equation indicates that all the β_{jj}s may be recovered. From here, the definition of diversion may be used to recover all the β_{ij}s.

[14]This assumption implies that customers substitute to products outside of the simulation in response to price increases by all products in the simulation at the same rate pre- and post-merger.

the 'mktElast' argument). The β_{ii}s are recovered from the fact that $\sum_j d_{ij} = 0$; customers must switch to a product included in the model.

If only a single product's own-price elasticity and ϵ is observed, then `pcaids` may be used in lieu of `aids` to calibrate the LA-AIDS parameters.[15]

Another distinguishing feature of the LA-AIDS model is that it does not require any information on product prices in order to simulate merger price effects. The LA-AIDS accomplishes this by using the supplied margin and revenue information to estimate B, but not α. There are however, a few drawbacks to not using pricing information. First, while merger-specific price *changes* may be calculated, pre- and post-merger price *levels* cannot. Second, welfare measures like compensating variation cannot be calculated. Prices are an optional input to `aids` and `pcaids`, and when they are supplied both price levels and welfare measures may be calculated.

Nested LA-AIDS

The nested LA-AIDS may be implemented using `pcaids.nests`.

By default, `aids` and `pcaids` assume that pre-merger, diversion occurs according to revenue share. While convenient, one potential drawback of this assumption is that diversion according to share may not accurately represent consumer substitution patterns. `antitrust` provides two ways to relax diversion according to share. First, both of these functions contain a 'diversions' argument that may be used to supply a $k \times k$ matrix of revenue diversions.

Alternatively, users can place the n products into $H \geq 2$ *nests*, with products in the same nest assumed to be closer substitutes than products in different nests.[16] This approach requires users to calibrate $\frac{H(H-1)}{2}$ nesting parameters, where each parameter measures the extent to which the diversion between any two products *in different nests* deviates from diversion according to share.[17] Accordingly, users must supply margin information for at least $\frac{H(H-1)}{2}$ products.[18]

---

[15]The main difference between `pcaids` and `aids` is that while `aids` requires users to supply revenue shares and at least two margins (or a single margin and the market elasticity) as inputs, `pcaids` requires the user to supply revenue shares, ϵ (using the 'mktElast' argument), and the own-price elasticity for one of the products (using the 'knownElast' argument). A value for 'knownElast' may be found by inverting the margin of a single-product firm. A value for 'mktElast' may be inferred from such sources as merging party documents, industry reports, and academic studies.

[16]No function in `antitrust` currently permits a hierarchy of nests.

[17]The nesting parameters are constrained to be between 0 and 1, where 1 means that diversion between nests occurs according to share. The diversion between two nests is assumed to be symmetric; the diversion from nest a to nest b is the same as the diversion from b to a.

[18]Note that these margins are in addition to the margin information that may be necessary to

1.2.4 Logit Demand

The Bertrand model with Logit demand may be implemented using the `logit` function.

Logit demand is based on a discrete choice model that assumes that each consumer is willing to purchase at most a single unit of one product from the n products available in the market. The assumptions underlying Logit demand imply that the probability that a consumer purchases product $i \in n$ is given by

$$s_i = \frac{\exp(V_i)}{\sum\limits_{k \in n} \exp(V_k)},$$

where s_i is product i's *quantity* share and V_i is the (average) indirect utility that a consumer receives from purchasing product i. We assume that V_i takes on the following form

$$V_i = \delta_i + \alpha p_i, \qquad\qquad \alpha < 0.$$

The Logit demand system yields the following own- and cross-price elasticities:

$$\epsilon_{ii} = \alpha(1 - s_i)p_i$$
$$\epsilon_{ij} = -\alpha s_j p_j$$

Logit demand has $n+1$ parameters to estimate (n δs and α) and up to $2n$ equations with which to estimate them (up to n complete FOCs and n choice probabilities). `calcSlopes` exploits this over-identification by employing a minimum distance algorithm to find the value for α that best satisfies all the FOCs for which there are data. The δs are then recovered from the choice probabilities.

One feature of the `logit` function is that the function allows users to specify whether or not consumers must purchase one of the n products sold in the market or whether consumers can choose to purchase an "outside" good. `logit` determines whether users wish to include an outside option by determining if the user-supplied quantity shares s_i sum to 1. If the shares sum to 1, then no outside good is included and by default δ_1 is normalized to 0.[19] Otherwise, an outside good is included whose δ

---

identify the elasticity of a single product ('knownElast').

[19]It can be shown that when there is no outside option in the Logit model, not all of the δs can be separately identified. Users can control which product's δ is normalized to 0 by setting `logit`'s 'normIndex' argument equal to the index (position) of the desired product.

is normalized to 0, price is set equal to 'priceOutside' (default 0), and whose share equals $s_0 = 1 - \sum_{i \in n} s_i$.[20]

Logit With Capacity Constraints

The capacity-constrained Bertrand model with Logit demand may be implemented using the `logit.cap` function.

The Logit Model with capacity constraints is calibrated by noting that in the pre-merger equilibrium, if product i is capacity constrained then $\frac{\partial q_i}{\partial p_j} = 0$ for all $j \in n$. This condition implies that an estimate of the price coefficient α may be obtained by starting with the FOCs in equation 1.1.1, deleting all rows pertaining to a capacity-constrained product and then for the remaining rows, zeroing out the appropriate elements of the Logit elasticity matrix E. A minimum distance estimator on the surviving FOCs is then employed to estimate the price coefficient. Once the price coefficient has been estimated, the technique outlined above may be used to uncover the vector of mean valuations.

Nested Logit

The Bertrand model with nested Logit demand may be implemented using the `logit.nests` function.

By construction, Logit demand assumes that diversion occurs according to quantity share. While convenient, one potential drawback of this assumption is that diversion according to share may not accurately represent consumer substitution patterns. One way to relax this assumption is to group the n products into $n > H \geq 2$ *nests*, with products in the same nest assumed to be closer substitutes than products in different nests.[21] `logit.nests`'s 'nests' argument may be used to specify a length-n vector identifying which nest each product belongs to.

The assumptions underlying nested Logit demand imply that the probability that a consumer purchases product i in nest $h \in H$ is given by

---

[20]Essentially 'priceOutside' controls how the mean valuations are scaled. Scaling is particularly important when computing compensating variation. See **?**, p.412 for further details.

[21]No function in `antitrust` currently permits a hierarchy of nests. Singleton nests (nests containing only a single product) are technically permitted, but their nesting parameter is not identified and is therefore normalized to 1.

$$s_i = s_{i|h} s_h,$$

$$s_{i|h} = \frac{\exp(\frac{V_i}{\sigma_h})}{\sum\limits_{k \in h} \exp(\frac{V_k}{\sigma_h})}, \qquad\qquad 1 \geq \sigma_h \geq 0$$

$$s_h = \frac{\exp(\sigma_h I_h)}{\sum\limits_{l \in H} \exp(\sigma_l I_l)}, \qquad\qquad I_h = \log \sum\limits_{k \in h} \exp\left(\frac{V_k}{\sigma_h}\right).$$

We assume that V_i takes on the following form

$$V_i = \delta_i + \alpha p_i, \qquad\qquad \alpha \leq 0.$$

The Nested Logit demand system yields the following own- and cross-price elasticities:

$$\epsilon_{ii} = [1 - s_i + (\frac{1}{\sigma_h} - 1)(1 - s_{i|h})]\alpha p_i,$$

$$\epsilon_{ij} = \begin{cases} -[s_j + (\frac{1}{\sigma_h} - 1)s_{j|h}]\alpha p_j, & \text{if } i, j \text{ are both in nest } h. \\ -\alpha s_j p_j, & \text{if } i \text{ is not in nest } h \text{ but } j \text{ is.} \end{cases}$$

Notice how these cross-price elasticities are identical to the non-nested Logit elasticities when products i, j are in different nests, but are larger when products i, j are in the same nests. This observation is consistent with the claim that products within a nest are closer substitutes than products outside of a nest.

In contrast to nested LA-AIDS, which must calibrate $\frac{H(H-1)}{2}$ nesting parameters, only H nesting parameters must be calibrated. By default, `calcSlopes` constrains all the nesting parameters to be equal to one another, $\sigma_h = \sigma$ for all $h \in H$. This reduces the number of parameters that need to be estimated to $n + 2$ (n δs, α, σ) which means users must furnish enough margin information to complete at least two FOCs. Setting `logit.nests`'s 'constraint' argument to FALSE causes the `calcSlopes` method to relax the constraint and calibrate a separate nesting parameter for each nest. Relaxing the constraint increases the number of parameters that must be estimated to $n + H + 1$, which means that users must furnish margin information sufficient to complete at least $H+1$ FOCs. Moreover, users must supply at least one margin per nest for each non-singleton nest. In other words, if nest $h \in H$ contains $n_h > 1$ products, then at least one product margin from nest h must be supplied.

Like `logit`, `logit.nests` also allows users to specify whether or not consumers must purchase one of the n products sold in the market or whether consumers can choose to purchase an "outside" good. This works almost the same in `logit.nests` as `logit`, except that when the sum of market revenue shares is less than 1, the outside good is placed in its own nest with its nesting parameter normalized to 1.

(Nested) Logit With Unobserved Outside Share

The Bertrand model with (nested) Logit demand and unobserved outside share may be implemented using the (`logit.nests.alm`) `logit.alm` function.

The Bertrand model with Logit demand described above assumes that when an outside good is included, its share is known. In some instances, however, users may find it difficult to reliably estimate the share of the outside good. The `logit.alm` function attempts to circumvent this issue by treating the share of the outside good as a nuisance parameter and using additional margin information to estimate that parameter.[22]

`logit.alm` accomplishes this by noting that the probability that a consumer purchases product $i \in n$ can be rewritten as

$$s_i = s_{i|I} s_I,$$
$$s_{i|I} = \frac{\exp(V_i)}{\sum_{k \in I} \exp(V_k)},$$
$$s_I = 1 - s_0,$$

where $s_{i|I}$ is product i's quantity share, conditional on a product being chosen from the set of inside goods I. This implies that

$$\sum_{k \in I} s_{k|I} = 1,$$

As in the Logit Model, we assume that V_i takes on the following form

$$V_i = \delta_i + \alpha p_i, \qquad\qquad \alpha \leq 0.$$

Likewise, the own- and cross-price elasticities may be rewritten as

---

[22]The outside good is a nuisance parameter because it is only needed to obtain estimates of the other demand parameters and is not used to solve for equilibrium prices.

$$\epsilon_{ii} = \alpha(1 - s_{i|I}(1 - s_0))p_i$$
$$\epsilon_{ij} = -\alpha s_{j|I}(1 - s_0)p_j$$

This version of the Logit model has $n + 2$ parameters to estimate (n δs, α, and s_0) and up to $2n$ equations with which to estimate them (up to n complete FOCs and n choice probabilities). `calcSlopes` exploits this over-identification by employing a minimum distance algorithm to find the values for α and s_0 that best satisfy all the FOCs for which there are data. The δs are then recovered from the choice probabilities.

Similar logic may be used to formulate the nested logit with unobserved outside share. Under the assumption that the outside good is the sole member of its own nest H_0, the probability that a consumer purchases product i in nest $h \in H \setminus H_0$ can be rewritten as

$$s_i = s_{i|h} s_h s_I,$$
$$s_I = 1 - s_0,$$

where $s_{i|h}$ and s_h are defined in the section on Nested Logit demand, and s_I is the unobserved probability that an inside good is selected. This version of the nested Logit model has $n + h + 2$ parameters to estimate (n δs, h nesting parameters, α, and s_0) and up to $2n$ equations with which to estimate them (up to n complete FOCs and n choice probabilities).

1.2.5 CES Demand

The Bertrand model with Constant Elasticity of Substitution (CES) demand may be implemented using the `ces` function.

Like the Logit, CES demand is based on a discrete choice model. However, CES differs from the Logit model in that under CES consumers do not purchase a single unit of a product but instead spend a fixed proportion of their budget on one of the n products available in the market.[23]

---

[23]Formally, each consumer chooses the product $i \in n$ that yields the maximum utility $U_i = \ln(\delta_i q_i) + \alpha \ln(q_0) + \epsilon_i$, subject to the budget constraint $y = p_i q_i + q_0$. Here, q_i is the amount of product i consumed by a consumer, δ_i is a measure of product i's quality, q_0 is the amount of the numeraire, y is consumer income, and ϵ_i are random variables independently and identically distributed according to the Type I Extreme Value distribution.

The assumptions underlying CES demand imply that the probability that a consumer purchases product $i \in n$ is given by

$$r_i = \frac{V_i}{\sum_{k \in n} V_k} \qquad \text{for all } i \in n,$$

where r_i is product i's *revenue* share and V_i is the (average) indirect utility that a consumer receives from purchasing product i. We assume that V_i takes on the following form

$$V_i = \delta_i p_i^{1-\gamma}, \qquad\qquad \gamma > 1.$$

The CES demand system yields the following own- and cross-price elasticities:

$$\epsilon_{ii} = -\gamma + (\gamma - 1)r_i$$
$$\epsilon_{ij} = (\gamma - 1)r_j$$

Functional form differences aside, one important difference between the CES and Logit demand systems is that the Logit model's choice probabilities are based on *quantity* shares, while the CES model's choice probabilities are based on *revenue* shares.

Like Logit demand, CES demand has $n + 1$ parameters to estimate (n δs and γ) and up to $2n$ equations with which to estimate them (up to n complete FOCs and n choice probabilities). `ces` exploits this over-identification by employing a minimum distance algorithm to find the value for γ that best satisfies all the FOCs for which there are data. The δs are then recovered from the choice probabilities.

`ces` also allows users to specify whether or not consumers must purchase one of the n products sold in the market or whether consumers can choose to purchase an "outside" good. `ces` determines whether users wish to include an outside option by determining if the user-supplied revenue shares r_i sum to 1. If the shares sum to 1, then no outside good is included and by default δ_1 is normalized to 1.[24] Otherwise, an outside good is included whose price and δ are normalized to 1, and whose share equals $r_0 = 1 - \sum_{i \in n} r_i$.

---

[24] It can be shown that when there is no outside option in the CES model, not all of the δs can be separately identified. Users can control which product's δ is normalized to 1 by setting `ces`'s 'normIndex' argument equal to the index (position) of the desired product.

In addition to specifying an outside option, `ces` has the 'shareInside' argument that may be used to specify the proportion of the representative consumer's budget that the consumer is willing to spend on the $n+1$ products that are within the market.[25] By default, 'shareInside' equals 1, which indicates that the customer spends her entire budget on the $n+1$ products within the market.

Nested CES

The Bertrand model with nested CES demand may be implemented using the `ces.nests` function.

Like the Logit, CES demand assumes that diversion occurs according to share.[26] While convenient, one potential drawback of this assumption is that diversion according to share may not accurately represent consumer substitution patterns. As with Logit demand, one way to relax this assumption is to group the n products into $H \geq 2$ *nests*, with products in the same nest assumed to be closer substitutes than products in different nests.[27] `logit.nests`'s 'nests' argument may be used to specify a length-n vector identifying which nest each product belongs to.

The assumptions underlying nested CES demand imply that the probability that a consumer purchases product i in nest $h \in H$ is given by

$$r_i = r_{i|h} r_h,$$
$$r_{i|h} = \frac{V_i}{I_h}, \qquad\qquad I_h = \sum_{k \in h} V_k$$
$$r_h = \frac{I_h^{\frac{1-\gamma}{1-\sigma_h}}}{\sum_{l \in H} I_l^{\frac{1-\gamma}{1-\sigma_l}}}.$$

We assume that V_i takes on the following form

$$V_i = \delta_i p_i^{1-\sigma_h},$$

---

[25] 1-'shareInside' equals the proportion of the representative consumer's income that is spent on all other products (i.e. the numeraire).

[26] CES assumes diversion according to revenue rather than quantity share.

[27] No function in `antitrust` currently permits a hierarchy of nests.

where $\sigma_h > \gamma > 1$ for all nests $h \in H$. The Nested Logit demand system yields the following own- and cross-price elasticities:

$$\epsilon_{ii} = - \sigma_h + (\gamma - 1)r_i + (\sigma_h - \gamma)r_{i|h},$$

$$\epsilon_{ij} = \begin{cases} (\gamma - 1)r_j + (\sigma_h - \gamma)r_{j|h} & \text{if } i, j \text{ are both in nest } h. \\ (\gamma - 1)r_j, & \text{if } i \text{ is not in nest } h \text{ but } j \text{ is.} \end{cases}$$

Like `ces`, `ces.nests` also allows users to specify whether or not consumers must purchase one of the n products sold in the market or whether consumers can choose to purchase an "outside" good. This works almost the same in `ces.nests` as `ces`, except that when the sum of market revenue shares is less than 1, the outside good is placed in its own nest with its nesting parameter normalized to 0.

By default, `calcSlopes` constrains all the nesting parameters to be equal to one another $\sigma_h = \sigma$ for all $h \in H$. This reduces the number of parameters that need to be estimated to $n + 2$ (n δs, α, σ) which means users must furnish enough margin information to complete at least two FOCs. Setting `ces.nests`'s 'constraint' argument to FALSE causes the `calcSlopes` method to relax the constraint and calibrate a separate nesting parameter for each nest. Relaxing the constraint increases the number of parameters that must be estimated to $n + H + 1$, which means that users must furnish margin information sufficient to complete at least $H + 1$ FOCs. Moreover, users must supply at least one margin per nest for each non-singleton nest. In other words, if nest $h \in H$ contains $n_h > 1$ products, then at least one product margin from nest h must be supplied.

1.2.6 Marginal Costs

If all n product margins are observed, estimating marginal costs can be accomplished by noting that $m_i \equiv \frac{p_i - c_i}{p_i}$ and using observed prices to calculate pre-merger marginal costs.

Rather than using observed margins to compute marginal costs, `antitrust` instead relies on the margins predicted by the Bertrand model. Rearranging the FOCs yields an expression for margins as a function of the demand parameters, product ownership, and revenue shares:

$$\hat{m}_{pre} = -((E'_{pre} \circ \Omega_{pre})^{-1} r_{pre}) \circ (\frac{1}{r_{pre}}),$$

where E_{pre}, r_{pre} are elasticities and revenues calculated from the assumed demand model, evaluated at observed prices.

The main advantage of using $\hat{m}_{pre}$ over m is that not all of the product margins must be observed in order to estimate marginal costs.[28] Once $\hat{m}_{pre}$ has been calculated, observed prices and the margin definition may be used to estimate pre-merger marginal costs.

Because `antitrust`'s Bertrand model assumes that marginal costs are constant, product i's post-merger marginal costs are equal to its pre-merger marginal costs, multiplied by $(1 + \Delta mc_i)$, the change in marginal costs due to any merger-specific efficiencies. All of the functions described above have a 'mcDelta' argument that allows users to specify a length-n vector of marginal cost changes.[29] By default, 'mcDelta' is equal to a length-n vector of zeros, indicating that the merger will not yield any efficiencies.

For the Bertrand model with capacity constraints, product margins for capacity-constrained products cannot be recovered from the first-order conditions.[30] Therefore, marginal costs for capacity-constrained products must be recovered from user-supplied margins and prices.

The `calcMC` method may be used to calculate pre- and post-merger marginal costs.

1.3 Simulating Merger Effects

For most of the demand systems included in `antitrust`'s Bertrand model, a closed-form solution in prices to the FOCs equation does not exist. We therefore employ the non-linear equation solver `BBsolve` from the `BB` package to find equilibrium prices. It is worth noting that the FOCs in equation 1.1.1 are necessary but not sufficient conditions for finding a price equilibrium to the Bertrand model. Unfortunately, there does not appear to be any theoretical result guaranteeing that, for many of the demand systems discussed here, there is a unique equilibrium to the Bertrand game in prices.[31] Practitioners sometimes address this problem by starting the

---

[28]Of course, enough margins must be observed to calibrate the demand parameters. For Log-Linear demand as well as Linear demand with a matrix of asymmetric slopes (B), all product margins must be supplied and $m = \hat{m}_{pre}$.

[29]Negative values for 'mcDelta' imply that a product's marginal cost will decrease, while positive values imply a price increase. Users will receive a warning if 'mcDelta' is supplied with positive values or if the values are greater than 1 in absolute value implying a cost change that is greater than 100%.

[30]To see why, note that equation 1.1.3 implies that if product i is capacity constrained pre-merger, then $\epsilon_{ij} = 0$ for all j. Since ϵ_{ij} is always multiplied by the margin of product i, that margin does not appear in the FOCs and is therefore not identified.

[31]To our knowledge, there is no theoretical result indicating that a unique Nash equilibrium in prices exists for most of the demand systems discussed here, i.e. when i) firms produce multiple

non-linear solver at different starting points in the price space, and seeing if these different initial values converge to distinct price equilibria. All of the constructor functions (e.g. `linear`,`loglinear`,`logit`) have a 'priceStart' argument that may be used to specify the non-linear solver's starting values. Moreover, many of these functions also include the 'isMax' argument, which when set equal to TRUE tests to see whether the candidate pre-merger and post-merger price equilibria identified by the non-linear solver are in fact (local) maxima.

`antitrust` users can also test the robustness of the predicted prices by modifying how the non-linear equation solver `BBsolve` used by most `antitrust` functions solves for the pre- and post-merger price equilibrium. Modifications to `BBsolve`'s default behavior may be accomplished by including `BBsolve` arguments in any of the `antitrust` functions described above.[32] See `BBsolve`'s help page for more information on how to modify `BBsolve`'s behavior.

The FOCs for the capacity-constrained Bertrand game (equation 1.1.3) suffer from an additional complication: the `max` function introduces a kink that can make it difficult for the non-linear equation solver to find equilibrium prices. **?**, p. 54 suggests replacing equation 1.1.3 with

$$FOC_i + q_i - t_i + \sqrt{FOC_i^2 + (q_i - t_i)^2} = 0, i = 1, \dots, n$$

which has the same roots as equation 1.1.3, but is smoother. The `calcPrices` method for all classes based on the capacity-constrained Bertrand Model use this smoothed system to solve for equilibrium prices.

In addition to computing pre- and post-merger equilibrium prices, `antitrust`'s Bertrand model contains methods that can compute many other features of the model. Below, we discuss a few of the methods that we expect users will find helpful. Table 7.2 provides a more extensive list of methods.

1.3.1 Summarizing Results

The `summary` method may be used to summarize the results of a merger between two firms for a given demand model. By default, the `summary` method reports pre- and post-merger equilibrium prices, *revenue shares*, weighted average compensating marginal cost reduction, and compensating variation.[33] *Quantity* shares rather than

---

products and ii) marginal costs are constant. The primary exception to this is linear demand.

[32]`linear`'s `calcPrices` method employs `constrOptim` rather than `BBsolve`.

[33]For some demand systems (e.g. Logit and CES), output shares as opposed to levels are reported. Compensating variation as well as equilibrium price levels for LA-AIDS models are reported

revenue shares may be reported by setting `summary`'s 'revenue' argument equal to FALSE. Likewise, levels, either in units or in revenues, rather than shares may be reported by setting `summary`'s 'shares' argument equal to FALSE. Calibrated demand parameters may be reported by setting `summary`'s 'parameters' argument equal to TRUE.The number of significant digits can be altered using the 'digits' argument.

In addition to printing the equilibrium price and output information to the screen, the `summary` method invisibly returns a matrix containing this information. Users can save this matrix to a new object for later use.

1.3.2 Plotting Results (experimental)

The `plot` method employs `ggplot` to plot pre- and post-merger product residual demand, marginal costs and equilibria. This method has a 'scale' argument, a number between 0 and 1 which controls how much of the demand curve above the equilibrium price and below marginal cost is plotted. The default for 'scale' is .1.

`plot` returns a `ggplot` object.

1.3.3 Simulating Price Effects With Efficiencies

Absent efficiencies, the Bertrand model with the demand systems described here will almost always produce a (possibly negligible) post-merger price increase among substitutes. These price increases, however, can be offset by merger-specific efficiencies that decrease the *incremental* costs of some of the merging firms' products.[34]

All the functions discussed above allow users to evaluate these efficiencies in two different ways. First, all of these functions contain the 'mcDelta' argument, which allows users to specify the proportional change in a product's marginal costs that may result from a merger. These cost changes are factored into the post-merger price equilibrium calculation made by the `calcPrices` method.

Second, users can call the `cmcr` method on the output of any of the functions described above. This method computes the compensating marginal cost reduction (CMCR) on the merging parties' products. CMCR is the percentage decrease in the marginal costs of the merging parties' products necessary to prevent a post-merger price increase. See the `cmcr` help page for further details.

_____

only if the user supplied pre-merger prices. Compensating variation is only reported for the Linear model if 'symmetry' equals TRUE.

[34]Costs that are not strictly increasing with a product's output (i.e. fixed or sunk costs) do not affect the price setting behavior of firms in a Bertrand pricing game.

1.3.4 Excluding Products From the Market (experimental)

By default, the Bertrand model calculates a merger's effects under the assumption that the acquisition does not change the set of products available to consumers. A merger's effects, however, may differ if the merger induces either the merging parties or another market participant to eliminate some products from their portfolio.

To accommodate the possibility that some products may be removed from the market following an acquisition, all the constructor functions described above have a 'subset' argument that allows users to specify a length-n logical vector that equals TRUE if a product should be included in the post-merger simulation and FALSE otherwise. By default, 'subset' is equal to a length-n vector of TRUEs.

1.3.5 Measuring Changes In Consumer Welfare

All of the demand models included in `antitrust`'s Bertrand model have a `CV` method which may be used to calculate compensating variation. Compensating variation is the amount of money needed to make a consumer as well off as they were before the merger increased prices. Table 7.3 lists the formula for calculating compensating variation for the demand models included in `antitrust`. The last column in this table indicates whether the formula for compensating variation returns compensating variation in levels (e.g. dollars) or as a percent of the representative consumer's total income.[35]

Compensating variation can be calculated only if i) the demand system is consistent with both consumer choice theory as well as the Bertrand model described above and ii) all the demand parameters can be estimated. As discussed earlier, the parameter restrictions necessary for the Log-linear demand system to satisfy consumer choice theory will typically not satisfy the parameter restrictions implied by the Bertrand model. Consequently, there is no `CV` method defined for the Log-linear demand system. Similarly, the `CV` method returns an error if Linear demand is calibrated without imposing symmetry and homogeneity of degree 0 in prices on the matrix of slope coefficients B (i.e. setting 'symmetry' equal to FALSE). Lastly, the `CV` method for LA-AIDS demand will return an error if LA-AIDS demand is calibrated without prices. This occurs because prices are needed to uncover estimates of the LA-AIDS demand intercepts, which are needed to compute compensating variation.

---

[35]The `CV` method for CES demand has a 'revenueInside' argument, which if set equal to the total revenue of all products included in the market, converts the percent to levels. Similarly, the `CV` method for LA-AIDS demand has a 'totalRevenue' argument, which if set equal to the representative agent's income (e.g. area GDP), converts the percent to levels.

Finally, it is worth noting that since none of the demand models included in `antitrust` contain income effects, it can be shown that compensating variation equals two other measures of consumer welfare: equivalent variation and consumer surplus.[36]

1.3.6 Defining Antitrust Markets

According to the 2010 Horizontal Merger Guidelines issued by the U.S Department of Justice (DOJ) and the Federal Trade Commission (FTC), the purpose of market definition is twofold:

> First, market definition helps specify the line of commerce and section of the country in which the competitive concern arises. In any merger enforcement action, the Agencies will normally identify one or more relevant markets in which the merger may substantially lessen competition. Second, market definition allows the Agencies to identify market participants and measure market shares and market concentration. [?, p. 7]

To assist users in identifying antitrust product and geographic markets, `antitrust` includes the `HypoMonTest` method. `HypoMonTest` assumes that i) firms are playing the differentiated Bertrand pricing game described earlier and ii) consumer demand is characterized by one of the demand systems described earlier, and then performs an implementation of the Hypothetical Monopolist Test described in the Guidelines for a set of products specified in `HypoMonTest`'s 'prodIndex' argument. [37]

Specifically, `HypoMonTest` first determines if 'prodIndex' contains at least one of the merging parties' products. If so, then by default `HypoMonTest` calls the `calcPriceDeltaHypoMon` method to find the profit-maximizing prices that the Hypothetical Monopolist would set on the products in 'prodIndex', holding the prices of all other products fixed at (predicted) pre-merger levels. `HypoMonTest` then compares the largest price change across the merging parties' products indexed in 'prodIndex' to the specified 'ssnip'. If this price change is greater than the specified 'ssnip', `HypoMonTest` returns TRUE.

---

[36]See ?.

[37]The Guidelines define the Hypothetical Monopolist Test for product market as positing

> ... a hypothetical profit-maximizing firm, not subject to price regulation, that was the only present and future seller of those products ("hypothetical monopolist") likely would impose at least a small but significant and non-transitory increase in price ("SSNIP") on at least one product in the market, including at least one product sold by one of the merging firms. For the purpose of analyzing this issue, the terms of sale of products outside the candidate market are held constant. [?, p. 9]

> The Guidelines describe a similar test for geographic market definition [?, p. 13]

Otherwise, `HypoMonTest` returns FALSE.

The Guidelines state that

> ... if the market includes a second product, the Agencies will normally also include a third product if that third product is a closer substitute for the first product than is the second product. The third product is a closer substitute if, in response to a SSNIP on the first product, greater revenues are diverted to the third product than the second product [**?**, p. 9].

To facilitate such comparisons, `antitrust`'s Bertrand model includes the `diversionHypoMon` method, which, for a set of products specified using the 'prodIndex' argument, returns the revenue diversion (as defined by equation 1.2.2)[38] matrix for *all* products included in the merger simulation (i.e. all products placed under the Hypothetical Monopolist's control as well as those outside of its control).

1.3.7 Simulating Merger Effects With Known Demand Parameters

Until now, most of the discussion has focused on how to recover demand parameters when users have information on shares, margins, and in most cases, prices. To accommodate known demand parameters (e.g. there is sufficient data to employ econometric methods to estimate demand parameters), antitrust contains the `sim` function. The `sim` function allows users to simulate price effects (or the output from any method listed in Table 7.2) from a merger under the assumption that firms are playing a Bertrand differentiated pricing game. `sim` requires users to specify a vector of market prices, demand form (either "Linear", "AIDS", "LogLin", "Logit", "CES", "LogitNests", "LogitCap", or "CESNests"), a list containing the known demand parameters, and pre- and post-merger ownership information. See the `sim` help page for further details.

1.4 Gotchas

This section alerts users to some instances where `antitrust` may produce seemingly surprising results, and provides some potential explanations for these behaviors.

---

[38]The Guidelines do not provide a formula for revenue diversion.

1.4.1 Market Definition

As discussed above, `HypoMon` method is a post-simulation command and therefore is run only after the user has assumed i) which firms (and products) are playing a differentiated Bertrand pricing game, and ii) the demand system. As a result, `HypoMon` and `diversionHypoMon` can never be applied to a set of products that includes any product excluded from the merger simulation.

1.4.2 Log-Linear Demand

`loglinear` always predicts no price effects for single-product firms in the market who are not party to the acquisition. This occurs because the FOC for a single-product firm producing i is $m_i = \frac{1}{\epsilon_{ii}} = \frac{1}{\beta_{ii}}$. Since the Bertrand model described earlier assumes constant marginal costs, a constant ϵ_{ii} implies that prices are constant as well.

Although a single-product non-merging party's prices will not change, its output will increase. This occurs because the acquisition will increase the price of the merging parties' products as well as the prices of multi-product non-merging parties. These price increases will entice some customers to switch towards the single-product firms' products, increasing their output.

1.4.3 LA-AIDS Demand

As discussed earlier, `aids` attempts to calibrate ϵ, the market elasticity parameter, by using the FOCs, LA-AIDS demand, and additional margins. For some combinations of margins and shares, however, this procedure can yield a very large market elasticity estimate, which in turn will yield small price effects from the merger. This issue appears to occur because the set of FOCs that are being used to calibrate this parameter are nearly co-linear. Issues arising from co-linearity can be often be remedied by supplying additional margin information, supplying margin information for products with disparate market shares, or supplying the market elasticity parameter directly.

2 The Cournot Quantity Game

Another model included in `antitrust` is the Cournot quantity game. This version of the game assumes that multi-plant firms with distinct, increasing marginal costs producing multiple products simultaneously set plant output for each product to maximize their profits. All firms producing a particular product are assumed to be undifferentiated. In this model, quantities are *strategic substitutes* in the sense that decreasing the quantity of one product causes some customers to switch to competing manufacturers, raising their quantities and profits. Ultimately, it is the magnitude of these lost sales that, at the margin, dissuades firms from reducing their output further .

Similar to the Bertrand model, mergers are modeled by assuming that the merging parties' plants are placed under common ownership, which allows the merged entity to *recapture* some of the sales that would otherwise be lost to competitors. As a result, the Cournot model (for some demand systems) predicts that absent any efficiencies affecting the marginal cost of production, the prices of all products produced by the merging parties will increase, possibly by a small amount.

Because all firms are manufacturing an identical version of the product, this version of the Cournot model allows firms post-merger to start or stop producing other products. This model does not allow for firms to engage in some forms of price discrimination.[1]

2.1 The Game

Suppose that there are K firms in a market, each producing a subset J_k of J products. Further, suppose that that each of the $k \in K$ firms manufactures its J_k products at n_k plants. Let $n = \sum_{k \in K} n_k$ denote the total number of plants producing any of the J products. The Cournot model assumes that firms simultaneously set the amount of each product produced at each plant in order to maximize their profits. This model also assumes that all firms can perfectly observe each others' quantities,

---

[1] In particular, this version of the Bertrand model does not accommodate non-linear pricing, such as is used in 2nd or 3rd degree price discrimination.

and costs, as well as the demand for each product.

Functions in `antitrust`'s Cournot model also adopt the additional assumption that each firm's plant has its own distinct marginal cost technology.

2.1.1 The Mathematical Model

Firm $k \in K$ chooses product output at each plant $\{q_j^r\}_{\substack{j \in j_k, \\ r \in n_k}}$, so as to maximize profits. Mathematically, firm k solves:

$$\max_{\substack{\{q_j^r\}_{j \in J_k}, \\ r \in n_k}} , \sum_{\substack{j \in J_k, \\ r \in n_k}} p_j q_j^r - \sum_{r \in n_k} c^r(q^r)$$

subject to

$$q_j^r \geq 0,$$
$$q^r = \sum_{j \in J_k} q_j^r$$

where p_j, the price sold of product i, is assumed to be a twice differentiable function of *all* firm quantities with $\frac{\partial p_j}{\partial q_j^r} < 0$ for all plants and firms . Likewise, additively separable plant variable costs c^r assumed to be twice differentiable with $\frac{\partial c^r}{\partial q_j^r} > 0$.

Differentiating profits with respect to each q_j^r yields the following first order conditions (FOCs):

$$\partial q_j^r \equiv p_j + \sum_{l \in n_k} q_j^l \frac{\partial p_j}{\partial q_j^r} - \frac{\partial c^r}{\partial q_j^r} = 0 \qquad \text{for all } j \in J_k, \ r \in n_k \qquad (2.1.1)$$

2.2 Calibrating Model Demand and Cost Parameters

The Cournot model can yield different equilibrium quantity and price predictions depending on 1) the curvature of plant variable costs and 2) the curvature of demand. `antitrust` allows users to explore the consequences of different cost and demand assumptions.

Currently, `cournot` contains two different ways to specify plant costs. First, users can set the 'cost' option equal to a n-length character vector whose values are either equal to "linear" for linear marginal costs ($\frac{\partial c^r}{\partial q_j^r} = 0.5\gamma_r \sum_{j \in n_r} q_j^r$) or "constant" for constant marginal costs ($\frac{\partial c^r}{\partial q_j^r} = \gamma_r$) . When the 'cost' option is employed, `cournot`'s `calcSlopes` method uses observed costs (implied by predicted margins and prices) and prices to calibrate plant-level cost parameters γ_r.

Alternatively, users can specify both plant level marginal and variable costs. Marginal costs may be specified by setting the 'mcfunPre' argument equal to a length n list of functions that each take as an input the vector of product quantities produced at a plant and then return the marginal cost associated with producing that level of output at that plant. Likewise, the 'vcfunPre' argument can be similarly used to specify plant-level variable costs. Note that when marginal costs and variable costs are specified in this manner, `cournot` makes no attempt to calibrate any parameters on the cost side.

Currently, `cournot` allows users to specify that product demand is either linear ($p_j = b_j + a_j \sum_{k \in n} q_j^k$) or log-linear ($\ln(p_j) = b_j + a_j \ln(\sum_{k \in n} q_j^k)$) Users can specify product demand by setting `cournot`'s 'demand' argument equal to a j-length character vector equal to either "linear" for linear demand or "log" for log-linear demand.

2.3 Simulating Merger Effects

2.3.1 Summarizing Results

The `summary` method may be used to summarize the results of a merger between two firms. By default, the `summary` method reports pre- and post-merger equilibrium prices and quantities for each product included in the simulation. Also reported is the compensating marginal cost reduction (CMCR) as well as the change in consumer surplus from the merger. Setting the 'market' argument equal to FALSE returns plant-level equilibrium price effects.

2.3.2 Simulating Price Effects With Marginal Cost Efficiencies

Absent efficiencies, the Cournot model with the demand systems described here will almost always produce a (possibly negligible) post-merger price increase among substitutes. These price increases, however, can be offset by merger-specific efficiencies

that decrease the *incremental* costs of some of the merging firms' products.[2]

All the functions discussed above allow users to evaluate these efficiencies in two different ways. First, all of these functions contain the 'mcDelta' argument, which allows users to specify the proportional change in a product's marginal costs that may result from a merger. These cost changes are factored into the post-merger price equilibrium calculation made by the `calcPrices` method. If changes in marginal costs are not expected to be proportional, then users can instead set the 'mcfun-Pre' and 'mcfunPost' arguments equal to lists containing functions that return each plants' pre- and post-merger marginal costs[3].

Second, users can call the `cmcr` method on the output of any of the functions described above. This method computes the compensating marginal cost reduction (CMCR) on the merging parties' products. CMCR is the percentage decrease in the marginal costs of the merging parties' products necessary to prevent a post-merger price increase. See the `cmcr` help page for further details.

2.3.3 Simulating Price Effects With Capacity Constraints

The Cournot model allows users to impose capacity constraints that may potentially change as a result of the merger. The "capacitiesPre" argument allows users to specify a vector of pre-merger plant-specific capacity constraints (default is **Inf**, or no constraint). The "capacitiesPost" argument allows users to specify a vector of post-merger constraints (default is "capacitiesPre").

2.3.4 Excluding Products or Plants

By default, the Cournot model calculates a merger's effects under the assumption that the acquisition does not change the set of products available to consumers or the set of plants that are in production. A merger's effects, however, may differ if the merger induces either the merging parties or another market participant to eliminate some products from their portfolio or to alter the mix of plants used in producing their products.

To accommodate the possibility that some products or plants may be removed from the market following an acquisition, all the constructor functions described above have a 'productsPre' and a 'productsPost' arguments that allows users to specify

---

[2]Costs that are not strictly increasing with a product's output (i.e. fixed or sunk costs) do not affect the price setting behavior of firms in a Bertrand pricing game.

[3]Users will also have to specify plant-specific variable costs as well using the 'vcfunPre' and 'vcfunPost' arguments.

an n by J logical matrix which equals TRUE if a product produced at a particular plant should be included in the merger simulation and FALSE otherwise. By default, 'productsPre' equals TRUE if the 'quantities' is not NA, and FALSE otherwise, while 'productsPost' equals 'productsPre'.

2.3.5 Measuring Changes In Consumer Welfare

The Cournot model has a `CV` method which may be used to calculate the change in consumer surplus from a merger.

2.3.6 Allowing For First-Mover Advantage

The package contains a `stackelberg` function that is similar to `cournot`, but also allows for one or more firms to be designated as output leaders for certain products. Output leaders are assumed to set their output levels first, and conditional on the leaders' output decisions, output followers (defined as everyone who is not a leader) chooses their levels of output. The `stackelberg` function contains four additional arguments: 'isLeaderPre' and 'isLeaderPost' for setting which firms are leaders and which are followers, and 'dmcfunPre' and 'dmcfunPost' for specifying the derivative of a plant's marginal cost (only necessary when marginal and variable costs are also specified).

3 Auction Models

Currently, `antitrust` has the ability to calibrate model parameters and simulate the effects of a merger in a:

- 2nd price procurement auction where the buyer sets a reserve price and capacity constrained bidders offer a single unit of a homogeneous product,

- 2nd score procurement auction with differentiated products, and a

- 1st score procurement auction with differentiated products.

In a 2nd *price* procurement auction, sellers submit their offers to supply a *homogeneous* product and the seller with the lowest offer wins but is paid the next lowest offer. 2nd price auctions have a number of interesting properties. In particular, it can be shown that:

- the dominant strategy in a 2nd price auction is for each seller to offer their product at marginal cost,

- the dominant strategy for a seller with a portfolio of products is to only offer the buyer the product in their portfolio with the lowest cost,

- the winner of the 2nd price auction is always the product with the lowest cost,

- the incremental surplus of the winning product, defined as the difference in cost between the least costly and next-most costly products, goes entirely to the seller,

- **absent efficiencies, horizontal mergers do no affect total surplus only consumer surplus.**

A 2nd *score* procurement auction generalizes a 2nd price auction by allowing sellers to offer *differentiated* products and allowing buyers to choose the product with the highest score. Instead of being paid the 2nd lowest cost, the seller with the highest score is paid the difference between the value of her product and the surplus (value less manufacturing cost) of the product with the next highest score. All the functions in `antitrust` assume that the non-price characteristics of a seller's products are fixed, which means that sellers strategies are restricted to i) which products to offer

a particular buyer and ii) what price to offer the products at.

All of the properties of a 2nd price auction generalize to a 2nd score auction. In particular:

- the dominant strategy in a 2nd score auction is for each seller to offer their product at marginal cost,

- the dominant strategy for a seller with a portfolio of products is to only offer the buyer the product in their portfolio with the highest surplus,

- the winner of the 2nd price auction is always the product with the highest surplus,

- the incremental surplus of the winning product, defined as the difference in surplus between the highest scored and next-most highest scored products, goes entirely to the seller,

- **absent efficiencies, horizontal mergers do no affect total surplus only consumer surplus.**

In contrast to a 2nd score procurement auction, sellers in a 1st score procurement auction simply receive their offer if they win. It can be shown that a first score auction where non-price product characteristics are fixed is equivalent to a Nash-Bertrand pricing game where buyers purchase only a single product.[1] Consequently, the models discussed in the chapter on the Bertrand Pricing Game, may also be used to explore the effects of mergers in first price auctions.

The remainder of this chapter explores `antitrust`'s implementation of 2nd score auctions

3.1 2nd Price Auction with Capacity Constraints

Here, we describe a 2nd price procurement auction where the buyer sets a reserve price and capacity constrained bidders offer a single unit of a homogeneous product.[2]. This auction model may be executed using `antitrust`'s `auction2nd.cap` function.

---

[1]See **?** for further details.

[2]**?** for further details.

3.1.1 The Game

Suppose that a buyer is interested in either purchasing a single unit of a homogeneous product from one of the K firms who supply the product, or supplying the product herself. Although the product is homogeneous, the cost of producing it is not: firm $k \in K$ uses constant marginal cost technology $c_k \geq 0$ to produce the product.[3] Likewise, suppose that the cost to the buyer of self-supply is c_0. Moreover, marginal costs are assumed to be private information, with the buyer and each firm believing that firm costs are independently drawn from the distribution $F(c)$ with support $(\underline{c}, \bar{c})$. Although marginal costs are constant, firms are assumed to be capacity constrained in the sense that all buyers and the bidder believe that a firm endowed with capacity $t_k \geq 0$ has a marginal cost c_k equal to the minimum of t_k cost draws from $F(c)$. Define $t = (t_1, \ldots, t_K)$ as the vector of firm capacities and $\bar{t} = \sum_{k \in K} t_k$ as total industry capacity.

The buyer is assumed to employ a 2nd price auction with a reserve price to determine who will supply the product. In this auction format, each firm submits a bid and the firm with the lowest bid that is also less than the buyer's reserve price wins the auction but is paid the 2nd lowest bid.[4] The buyer's reserve reflects the fact that the buyer has the ability to *self-supply* at cost c_0, and so is only willing to purchase the product from a bidder whose bid is is less than c_0.

It can be shown that in a 2nd price auction, each firm has a weakly dominant strategy to bid it's marginal cost. **?** also show that

- the probability that firm k's cost draw is less than c, equals

$$G(c; t_k) \equiv 1 - [1 - F(c)]^{t_k}, \qquad (3.1.1)$$

- the probability that firm k wins the auction for a given reserve price r, is

$$s_k(r; \bar{t}) \equiv \frac{t_k}{\bar{t}} G(r; \bar{t}), \qquad (3.1.2)$$

- the probability that firm k wins conditional on I, the event that *some* firm wins, is

$$s_{k|I}(t) \equiv \frac{t_k}{\bar{t}}, \qquad (3.1.3)$$

---

[3] Throughout, we abuse the notation slightly by treating variables like K as both the set of firms as well as the number of firms.

[4] If the second lowest bid is greater than the reserve price, the winner receives the reserve.

- firm k's expected producer surplus is

$$\Pi_k(r; t) \equiv \int_{\underline{c}}^{r} \{[1 - F(c)]^{\bar{t} - t_k} - [1 - F(c)]^{\bar{t}}\} dc, \qquad (3.1.4)$$

- firm k's expected cost is

$$E[c_k] \equiv t_k \int_{\underline{c}}^{r} c f(c)[1 - F(c)]^{\bar{t} - 1} dc, \qquad (3.1.5)$$

- firm k's expected price is

$$E[p_k] = \Pi_k(r; t) + E[c_k] \qquad (3.1.6)$$

The buyer sets her reserve price r so as to minimize her expected costs (EC):

$$EC(r, t) = c_0 Pr(\min_{k \in K}\{c_k\} > r) + p(r, t) Pr(\min_{k \in K}\{c_k\} \leq r)$$
$$= c_0(1 - G(r; \bar{t})) + p(r, t) G(r; \bar{t})$$

where $p(r, t)$ is the expected price paid by the buyer conditional on the buyer purchasing from some firm, i.e.

$$p(r, t) \equiv \frac{\int_{\underline{c}}^{r} c \, dG(c; \bar{t})}{G(r; \bar{t})} + \frac{\sum\limits_{k \in K} \Pi_k(r; t)}{G(r; \bar{t})}$$

which is minimized at

$$c_0 = r^* + \frac{\sum\limits_{k \in K} ([1 - F(r^*)]^{\bar{t} - t_k} - [1 - F(r^*)]^{\bar{t}})}{\bar{t} f(r^*)[1 - F(r^*)]^{\bar{t} - 1}} \qquad (3.1.7)$$

Note that this last equation implies that the buyer has an incentive to set its optimal reserve below its self-supply cost.

3.1.2 Calibrating Model Parameters

In this model, we will assume that the user observes firm capacities, *some* price and margin information, and possibly the auction reserve price or the proportion of times a buyer opts to self-supply. The unknowns are the seller cost distribution $F(c)$, the buyer's cost of self-supply c_0, and if not observed, the auction's reserve price.

Selecting a Cost Distribution

It can be shown that under this models' assumptions, it must be the case that $F(c) = 1 - \exp k(c)$. Common distributions that satisfy this functional form are the Uniform, Exponential, Weibull, Gumbel, and Frechet. Table 3.1 summarizes these distributions.

Name	Bounds	Location Parameter	Scale Parameter	Shape Parameter	# Params
Uniform	$[\underline{c}, \overline{c}]$				2
Exponential	$[0, \infty)$		X		1
Weibull	$[0, \infty)$		X	X	2
Gumbel	$(-\infty, \infty)$	X	X		2
Frechet	$[Location, \infty)$	X	X	X	3

Table 3.1: Five Cost Distributions that satisfy this model's assumptions. An 'X' indicates a parameter that must be calibrated.

It is important to note that depending upon the parameters, these distributions can be shaped very differently from one another, which can yield different simulated merger effects. In general, distributions with more parameters can assume a greater range of shapes than distributions with fewer parameters. This flexibility comes at a cost; the user must supply additional information in order to calibrate these additional parameters.

Identifying Moments

Once a distribution has been selected, the distribution parameters as well as the reserve price (if not supplied) must be calibrated from user-supplied prices and margins. To accomplish this, first let $\hat{p}_k$ and $\hat{c}_k$ denote firm k's *ex ante* price and marginal cost, conditional on k winning. Equations 3.1.5, 3.1.6, and 3.1.2 together imply that $\hat{p}_k = \frac{E[p_k]}{s_k(r;\bar{t})}$ and that $\hat{c}_k = \frac{E[c_k]}{s_k(r;\bar{t})}$ Letting $\overline{p_k}$ and $\overline{c_k}$ denote firm k's average price and marginal cost yields the following moments:

$$\overline{p_k} = \hat{p}_k$$
$$\overline{c_k} = \hat{c}_k$$

If in addition, s_0, the proportion of buyers who opt to self-supply is observed, then the following moment is also employed:

$$s_0 = 1 - G(r; \bar{t})$$

Note that in order to calibrate the model parameters, the number of moments with user-supplied information must be greater than the number of distribution parameters plus the reservation price (if not supplied).

3.1.3 Simulating Merger Effects (experimental)

Summarizing Results

The auction model's `summary` method may be used to summarize the results of a merger between two firms for a given cost distribution. By default, the `summary` method reports pre- and post-merger equilibrium prices and *quantity* shares, conditional on *some* firm winning the auction. The unconditional pre- and post-merger equilibrium prices and quantities may be reported by setting `summary`'s 'exAnte' argument equal to TRUE. Calibrated distribution parameters may be reported by setting `summary`'s 'parameters' argument equal to TRUE. The number of significant digits can be altered using the 'digits' argument.

Simulating Price Effects With Efficiencies

By default, `auction2nd.cap` assumes that a merger places the parties' capacities under common ownership and that the capacities of all other firms in the market are unaffected by the acquisition. Both of these assumptions may be modified with the 'mcDelta' argument, which allows users to specify the proportional changes in each firm's capacities due to the merger.

3.2 2nd Score Auction with Differentiated Products

Here, we describe a 2nd score procurement auction where the features of the products – including cost – are both horizontally and vertically differentiated. This section is largely based on **?** and **?**. This auction model may be executed using `antitrust`'s `auction2nd.logit` function.

3.2.1 The Game

Suppose that a buyer is interested in either purchasing a single unit of a differentiated product from one of K firms, each of whom manufacture $J_k, k \in K$ variants of the product, or in supplying the product herself. Let $J = \bigcup_k^K J_k$ denote the set of all

products produced by any of the K firms. Not only do each of the variants have different characteristics, the cost of producing these products may also differ. In particular, suppose that variant $J_k, k \in K$ uses constant marginal cost technology $c_{J_k} \geq 0$ to produce the product.[5]

Let the utility that a buyer receives from product $j \in J$ with offer p_j equal to

$$V_j(p_j) = \overline{V}_j(p_j) + \epsilon_j$$
$$\overline{V}_j(p_j) = \delta_j + \alpha p_j, \qquad\qquad\qquad \alpha < 0,$$

where, $\overline{V}_j$ represents an index of "vertical" quality differentiation that decomposed into a non-offer component δ_j and an offer component αp_j. In addition, ϵ_j represents a buyer-specific idiosyncratic shock to utility (i.e. "horizontal" quality differentiation). Without loss of generality, We will assume that these shocks are independently drawn from the distribution F, with mean 0 and variance 1.[6] We will also assume that at the start of the auction, ϵ_j is known to j's manufacturer but is not known to any other sellers.

The buyer is assumed to employ a 2nd score auction to determine which product she will purchase. In this auction format, each firm submits an offer and the firm with the highest score (which may be the buyer if the buyer self-supplies) wins the auction but pays the difference between the value of the highest scoring product and the value of the next-highest scoring product, less that seller's offer on that product. It can be shown that it is a *dominant strategy for each firm to offer the product with the highest surplus in its portfolio to the buyer at cost.*

Let $z_A = \max_{k \in A} V_k(c_k)$ denote the the maximum surplus available from any product $k \in A \subseteq J$. It can be shown that the following must hold when firm $k \in K$ wins with variant $j \in J_k$

- the probability that j wins is

$$s_j \equiv 1 - Pr(V_j < z_{k \notin J_k}); \qquad\qquad (3.2.1)$$

- the expected price conditional on j winning is

$$E[p_j | j \text{ wins}] = c_j + E[z_J | j \text{ wins}] - E[z_{k \notin J_k} | j \text{ wins}]; \qquad (3.2.2)$$

---

[5]Throughout, we abuse the notation slightly by treating variables like K as both the set of firms as well as the number of firms.

[6]Equivalently, $-\frac{V_j(p_j)}{\alpha}$ is a random variable with location $-\frac{\overline{V}_j(p_j)}{\alpha}$ and scale $-\frac{1}{\alpha}$.

- subtracting c_j from both sides of 3.2.2 yields the expected profit margin conditional on j winning:

$$E[m_j | j \text{ wins}] = E[z_J | j \text{ wins}] - E[z_{k \notin J_k} | j \text{ wins}] \qquad (3.2.3)$$

Merger analysis

Suppose that firm $k \in K$ acquires a portfolio of products J_r produced by firm $r \in K$.[7]

Retention of all products. If post-merger k does not discontinue any of r's products, then it can be shown that it is still a dominant strategy for k to only offer a buyer the highest surplus product, and that consequently, the acquisition does not affect which product will ultimately be selected *ex-ante*. Consequently, the merger does not the decrease *ex-ante* output and therefore leaves total surplus unchanged. The merger, will, absent variable cost efficiencies lead to a price increase that exactly equals the decrease in consumer surplus. The *ex-ante* magnitude of the price increase (and corresponding decrease in consumer surplus) when $j \in J_k \cup J_r$ win the auction is:

$$E[\Delta p_j | \text{j wins}] = E[m_j^{post} | \text{j wins}] - E[m_j^{pre} | \text{j wins}]. \qquad (3.2.4)$$

The expected change in consumer surplus– which, absent efficiencies also equals the weighted average price effect– may be calculated by taking the share- weighted average of equation 3.2.4:

$$E[\Delta p] = \sum_{j \in J_k \cup J_r} s_j E[\Delta p_j | j \text{ wins}] \qquad (3.2.5)$$

where s_j is given by 3.2.2

3.2.2 Calibrating Model Demand and Cost Parameters

For this model, we will assume that users observe firm shares, prices, and *some* margin information. From the user's perspective, the unknowns are the buyers'

---

[7]Note that k could either acquire r's entire product line or just a subset.

distribution of valuations (F), the constant marginal costs (c_j) and the components of a buyer's mean valuation (i.e. the δ_js and α).

In order to calibrate this model with the limited information available, we will need to assume a functional form for F. Currently, `antitrust` assumes that F follows a Gumbel distribution. Under this distributional assumption,

- the probability that j wins is

$$s_j = \frac{\exp(\overline{V}_j(c_j))}{\sum\limits_{k \in J} \exp(\overline{V}_k(c_k))} \tag{3.2.6}$$

- the expected value of the maximum for products in $A \subseteq J$ is

$$E[z_A] = -\alpha^{-1} \log \sum\limits_{j \in A} \exp(\overline{V}_j(c_j)) \tag{3.2.7}$$

First, note that for product $h \neq j$, equation 3.2.6 indicates

$$\log s_j - \log s_k = \overline{V}_j(c_j) - \overline{V}_h(c_h) \tag{3.2.8}$$
$$= \delta_j - \delta_h - \alpha(c_j - c_h) \tag{3.2.9}$$

Equation 3.2.6 implies that at least one of the V_js is not separately identified. Therefore, one product must be selected as the *numeraire* so that all the other valuations are *relative* to the normalized product.[8]

Next, substituting equation 3.2.7 into equation 3.2.3 yields the following closed form expression for margins:

$$E[m_j | V_j(c_j) > z_{k \notin J_k}] = -\frac{1}{\alpha \sum\limits_{r \in J_k} s_r} \log \left(\frac{1}{1 - \sum\limits_{r \in J_k} s_r} \right) \tag{3.2.10}$$

Equations 3.2.9 and 3.2.10 form the basis of the calibration strategy. All the model parameters may be calibrated with just a single margin and market shares. Product prices only need to to be supplied for products whose marginal costs are assumed to change post-merger (i.e. products for which 'mcDelta' is not equal to 0).

---

[8]Specifically, suppose k is the *numeriare* product, and that $V_k = \delta_k + \alpha p_k$. When supplied 'shares' sum to 1, the first product is by default set to be the numeraire product. This may be changed with the 'normIndex' option.

For example, suppose that the margin for product $j \in J_k$ is known. Equation 3.2.10 implies that α may be recovered using only j's margin as well as the sum of the shares for all of firm j's products. This estimate of α, in conjunction with equation 3.2.10 may then be used to estimate margins for all the other products in the market. Absent post-merger cost changes, these margin estimates, along with equation 3.2.9, may then be used to estimate each products' non-offer mean valuations (δs).

When post-merger cost changes are included in the model, it becomes necessary to estimate marginal costs, *but only for the products whose costs are changing due to the merger.*[9] This is accomplished by including price information for these products. Products whose costs are not assumed to change do not require pricing data (i.e. 'prices' may be NA for products where 'mcDelta' is 0).

In some instances, there may be more than one margin available. In these cases a minimum distance algorithm is used to find the value of α that best satisfies all the margin equations (3.2.10) for which there is data.

3.2.3 Simulating Merger Effects

Summarizing Results

The `summary` method for `Auction2ndLogit` is similar to the method described in section 3.1.3. The main difference is that by default, this method reports changes in levels rather than percentages and output in quantity shares rather than in revenue shares. These alterations were made because prices are not a pre-requisite for this model, but when prices are not supplied, only level changes are identified. Regardless, these defaults can be changed by appropriately setting the methods 'revenue' and 'levels' arguments.

Simulating Price Effects With Efficiencies

`auction2nd.logit`'s 'mcDelta' argument may be used to specify the anticipated proportional change in marginal costs that will likely occur from a merger. See `auction2nd.logit`'s help page for further details.

---

[9]This occurs because in a 2nd score auction, it is a dominant strategy to bid costs. Consequently, if product j's costs do not change, then V_j is the same for product j both pre- and post-merger. This is in contrast to the Nash-Bertrand pricing game described in Section 1.2.4, where a firm's pricing strategy for product j depends on the pricing behavior of all other products, making it necessary to use pricing information to decompose V_j into its price and non-price components.

Excluding Products From the Market

`auction2nd.logit`'s 'subset' argument may be used to simulate a merger's effect when a product is assumed to be eliminated from the market post-merger. See `auction2nd.logit`'s help page for further details.

Measuring Changes In Consumer Welfare

The `CV` method may be used to calculate compensating variation from a merger. This method will return 'NA' if pre-merger prices were not supplied for all products included in the simulation.

Calibrating Unobserved Outside Share

`auction2nd.logit.alm` uses the calibration strategy similar to the one discussed in section 1.2.4 to calibrate the unobserved share of the outside good.

4 Other Tools

For some acquisitions, there may be insufficient information available to use any of the merger simulation functions described above. In these instances, if information is available on the merging parties' products, then it may still be possible to calculate measures that can help inform users about the effects of the merger.

4.1 CMCR

One such measure, discussed above, is compensating marginal cost reduction (CMCR). CMCR measures the change in the marginal cost of the merging parties' products needed to offset the price increase following the merger. CMCR may then be compared to the merger's efficiencies in order to determine whether or not the merger will lead to a price increase.

`cmcr.bertrand` may be used to compute CMCR under the assumption that the merging parties are playing the Bertrand pricing game described earlier. The formula for $CMCR_{Bertrand}$ in matrix notation is:

$$CMCR_{Bertrand} = (m_{post} - m_{pre}) \circ \frac{1}{1 - m_{pre}},$$

where m_{pre} is a vector of observed pre-merger product margins for each of the merging parties' products. m_{post}, post-merger margins evaluated at pre-merger prices, may be found using

$$m_{post} = (B_{post})^{-1} \left(\frac{diag(\Omega_{post})}{diag(\Omega_{pre})} \circ B_{pre} m_{pre} \right)$$

$$B_s = D_{pre}^q \circ ((1/p_{pre}) p'_{pre}) \circ \Omega_s, \qquad s \in \{pre, post\},$$

where D_{pre}^q is a matrix of pre-merger quantity diversion ratios for the merging parties' products whose i, jth element is the quantity diversion from product i to product j, p_{pre} is a vector of pre-merger prices for the merging parties' products, and Ω_s is a matrix of either pre- or post-merger ownership shares (typically equal to 1). Note

that this formula requires users only to supply price and margin information for all of the merging parties' products, as well as diversion information between all of the merging parties products.

`cmcr.cournot` may be used to compute CMCR under the assumption that the merging parties are playing a Cournot quantity-setting game where each party produces a single product. The formula for $CMCR_{Cournot}$ is:

$$CMCR_{Cournot} = \frac{2s_i s_j}{\epsilon(s_i + s_j) - (s_i^2 + s_j^2)},$$

where i and j index the merging parties products and ϵ is the equilibrium elasticity of industry demand. This function requires users to supply information on the merging parties' quantity shares as well as an estimate of the market elasticity. Under the assumption that each firm produces a single product, it can be shown that $\epsilon = \frac{s_i}{m_i}$ for all products i. Hence, only a single margin is needed to recover an estimate of ϵ.

The main drawback to using CMCR is that CMCR yields only the reduction in marginal costs needed to prevent a price increase; it does not provide any information on how much prices would increase if the efficiencies from the merger are less than CMCR. Likewise, CMCR cannot be used to draw inferences about price effects if some of the merging parties' products are expected to yield efficiencies that are larger than CMCR, while others are expected to yield efficiencies that are smaller than CMCR.

4.2 Generalized Pricing Pressure

Another measure included in `antitrust` is Generalized Pricing Pressure (GePP).[1] GePP measures how a merger would affect the merging parties' incentives to change the prices of their products, after accounting for any merger-specific efficiencies. The GePP for the merging parties' products may be written as

$$GePP = (B_{post}m_{post})/diag(\Omega_{post}) - (B_{pre}m_{pre})/diag(\Omega_{pre})$$

where B_{pre}, B_{post} and m_{pre} are the same as in CMCR,and m_{post} are pre-merger margins that incorporate anticipated cost reductions. GePP predicts that the acquiring firm will have an incentive to raise the price of merging party product i when $GePP_i > 0$.

---

[1]GePP (**?**) extends Upward Pricing Pressure (UPP) – introduced in **?**– to accommodate multi-product merging firms.

`upp.bertrand` may be used to compute GePP under the assumption that the merging parties are playing the Bertrand pricing game described earlier. Like `cmcr.bertrand`, this function requires users to supply price and margin information for all of the merging parties' products, as well as diversion information between all of the merging parties products. Users can also supply a vector of merger-specific efficiencies to `upp.bertrand`'s 'mcDelta' argument (default is 0, which assumes no efficiencies). These efficiencies should be expressed as the percentage decrease in the merging parties' marginal costs.

4.3 HHI

The 2010 Horizontal Merger Guidelines state that

> The Agencies often calculate the Herfindahl-Hirschman Index ("HHI") of market concentration. ... The higher the post-merger HHI and the increase in the HHI, the greater are the Agencies' potential competitive concerns and the greater is the likelihood that the Agencies will request additional information to conduct their analysis." [**?**, p. 18].

`antitrust` contains the `HHI` function to compute the HHI for a specified set of products. `HHI` also allows users to compute the Modified HHI (MHHI) that may be used to account for partial firm ownership– where one firm receives a share of the profits from another firm's product, as well as partial control– where one firm has the (partial) ability to control how much of another firm's product is produced.

Part II

Coordinated Effects

5 A Collusion Game with Bertrand Reversion as Punishment

The basic idea behind collusion is that when firms interact with one another repeatedly – either across different product or geographic markets, or over time – firms may find a way to reach an agreement that reduces the extent to which they compete with one another. This agreement may be *tacit* in the sense that a firm can coordinate its behavior with that of its rivals by for example, first unilaterally adopting a strategy that would be mutually beneficial to itself and its rivals, and then observing whether its rivals subsequently choose to either *cooperate* by adopting a mutually beneficial strategy or *defect* by adopting a strategy that is only beneficial to themselves. Colluding firms may also attempt to *punish* defecting rivals in order encourage these rivals to cooperate rather than defect. A number of factors can influence a firm's decision to collude including:

- how willing a firm is to sacrifice short term for long term gain,
- the ease with which a firm can implement a mutually beneficial strategy,
- the ease with which a firm can observe its rivals' response,
- the costs associated with punishing a defecting rival.

Our aim here is to explore how a horizontal merger might affect the incentives of firms to collude. In particular, a horizontal merger can facilitate collusion if a firm with an incentive to collude acquires a *maverick*, or a firm without such incentive. By doing this, the acquiring firm makes it easier to sustain cooperation among some of the remaining firms. On the other hand, a horizontal merger can retard collusion if after the acquisition, a firm believes that it can make greater surplus by not cooperating with some of the remaining firms. The goal of this model is to provide one method for exploring which of these conflicting incentives ultimately prevails.

5.1 The Game

Suppose that there are K firms in a market, and that each of the $k \in K$ firms produces n_k products.[1] Let $n = \sum_{k \in K} n_k$ denote the number of products sold by all K firms. These firms are playing an infinitely repeated super-game, where in each period of the game the K firms play the Bertrand pricing game described in section 1.1. Mathematically, firm k solves:

$$\max_{\{p_{kt}\}_{t=1}^{\infty}} \sum_{t=1}^{\infty} \sum_{i=1}^{n} \tau_i^{t-1} \Pi_{ikt}^b,$$

where p_{kt} is the vector of prices set on all of k's products in period t; $0 < \tau_i < 1$ is a product-specific and time-invariant discount factor reflecting the rate at which an owner of i is willing to forgo earning surplus today on i for future surplus; and Π_{ikt}^b is the surplus earned by firm k on product i in period t. Mathematically,

$$\Pi_{ikt}^b = \omega_{ik}(p_{it} - c_i)q_{it},$$

where as in the Bertrand Model, ω_{ik} is the share of product i's surplus earned by firm k absent collusion, so that $\sum_{k \in K} \omega_{ik} \leq 1$. q_{it}, the quantity sold of product i at time t, is assumed to be a twice differentiable function of *all* product prices. Note that ω_{ik} and marginal cost c_i are assumed to be constant over time.

Further, suppose that some subset of firms $Z \subseteq K$ wish to collude such that each firm $z \in Z$ selects a subset $J_z \subseteq n_z$ of their products over which to jointly maximize surplus. We assume that firms within this Z coalition cannot make side-payments to one another. In other words, firms inside the coalition only earn surplus on the products that they produce.[2] Let Π_{jzt}^c denote the surplus that firm $z \in Z$ earns from selling product $j \in J_z$ at the jointly surplus maximizing price in period t, and let Π_{lzt}^c denote the surplus that firm z earns from product $l \notin J_z$ in period t. Note that for a given product $l \notin J_z$, the profits that z earns on l under collusion (Π_{lzt}^c) will typically differ from the profits that z earns under Bertrand reversion (Π_{lzt}^b), even though in both instances l is not included in the coalition.

Following **?** and **?**, we assume that firms within Z all attempt to enforce the coalition by playing Grim Trigger strategies where each firm in Z colludes until they observe

---

[1] Throughout, we abuse the notation slightly by treating variables like K and n_k as both the set of firms as well as the number of firms.

[2] Future versions of `antitrust` may extend this model to allow for side payments.

a single defection from any other firm in Z, and then "punishes" by forever reverting the prices of its products back to those from the Bertrand game.

All that remains is to describe a firm's surplus from defecting from the coalition. It is important to note that under the assumptions that all firms are playing Grim Trigger strategies *and* at most one firm can defect in a turn, i) only a single firm will be able to earn the defector's surplus, and ii) that firm will only be able to earn the defector's surplus for a single period. Here, we assume that when a firm defects it sets the prices of its products to maximize only the surplus of its products, conditional on every other firm in Z setting prices to maximize the coalition's surplus. Let Π_{jzt}^d denote the surplus that firm $z \in Z$ earns on product j in period t from setting its products' prices to only maximize the surplus of its products, subject to the other firms in Z setting prices to maximize joint surplus.

5.1.1 The Incentive to Collude Under Grim Trigger

In this model, Firm z has an incentive to collude with the other firms in the Z coalition only if the discounted value of the surplus it gains from colluding is greater than the discounted value of the surplus it receives from defecting from coalition prices for a single period, followed by the discounted value of surplus it receives from all firms earning surplus from the Bertrand game for all subsequent periods. Mathematically, this condition can be written as:

$$\underbrace{\sum_{t=1}^{\infty}\sum_{j=1}^{n}\tau_j^{t-1}\Pi_{jzt}^b}_{\text{collusion surplus}} \geq \underbrace{\sum_{j=1}^{n}\Pi_{jzt}^d}_{\text{defection surplus}} + \underbrace{\sum_{t=2}^{\infty}\sum_{j=1}^{n}\tau_j^{t-1}\Pi_{jzt}^b}_{\text{reversion surplus}}$$

Since we are assuming that outside of the merger firms cannot enter or exit, or add or remove products, equilibrium prices and quantities are the same across all periods, and the above simplifies to

$$\sum_{j=1}^{n}\frac{\Pi_{jzt}^c}{1-\tau_j} \geq \sum_{j=1}^{n}\Pi_{jzt}^d + \sum_{j=1}^{n}\frac{\tau_j\Pi_{jzt}^b}{1-\tau_j} \tag{5.1.1}$$

5.2 Comparing Incentives to Collude and Defect Under Grim Trigger (experimental)

Inequality (5.1.1) indicates that five pieces of information must be identified in order to determine if firm $z \in Z$ has an incentive to collude under the model here: the set of products involved in the Z coalition, the discount rates τ_j for all products produced by firms with at least one product in the coalition (i.e for each $j \in n_z$); the surplus firm z earns from each product $j \in n_z$ from colluding (Π^c_{jzt}) in period t, the surplus firm z earns from each product $j \in n_z$ upon defecting from the Z coalition (Π^d_{jzt}) in period t, and the period t surplus a firm earns from each product $j \in n_z$ after all firms have "punished" by reverting to Bertrand pricing Π^b_{jzt}.

The `calcProducerSurplusGrimTrigger` method assumes that the user has supplied the indices for all products involved in the coalition to the 'coalition' argument, as well as the discount rates τ_j to the 'discount' argument. This method then computes 'preMerger' product profits Π^c_{jzt}, Π^d_{jzt}, and Π^b_{jzt} for products owned by firms participating in the 'coalition' and then returns a data frame containing the inputted discount rates, these surplus calculations, and a field labeled "IC" which equals TRUE if inequality (5.1.1) holds and FALSE otherwise. Note that "IC" is a firm-level calculation whose value will be replicated across all products owned by the same firm.

`calcProducerSurplusGrimTrigger` is designed to be run twice: once with 'preMerger' equal TRUE in order to evaluate whether, under this model's assumptions members of the specified coalition have an incentive to collude; and a second time with 'preMerger' equal to FALSE in order to evaluate how a merger affects these incentives.

Part III

Under The Hood

6 Getting Help

A manual page has been written for each class, method, and function contained in `antitrust`. These pages describe the relevant object, its inputs and outputs, and typically contain at least one example of how it is used. See the `help` function for assistance on how to access these pages.

In addition to the manual pages, `R`'s S4 system includes a number of ways to investigate the properties of the classes and methods contained in `antitrust`. To learn more about a particular instance of an `antitrust`class, use the `str` command. To learn more about the class itself (e.g. class slots, who its parent and child classes are) use the `showClass` function. To show which methods are defined for a class, or which classes have a particular method, use the `showMethods` function. To see how a method has been defined, use the `getMethod` function.

7 Modifying and Extending `antitrust`

`antitrust` was written using `R`'s S4 object-oriented class system. Figure 7.1 displays the relationships between each parent-child class. The figure indicates that the *Antitrust* class is the main class. Indeed, every effort has been made to include in the *Antitrust* class all slots that are common to its child classes as well as all common methods.

7.1 The Bertrand Model

Figure 7.1 also reveals that the *Bertrand* class is the parent class for all models related to the Bertrand Pricing game. Note that each of the *Bertrand* class's child classes are named after a demand model included in `antitrust`. These classes are grouped into two branches: demand systems based on the representative consumer's value function (the *Logit* branch) and demand systems based on the representative consumer's expenditure function (the *Linear* branch).

Each `antitrust` class named after a demand model has a similarly named constructor function associated with it. For example, the *Linear* class has the `linear` constructor function associated with it. The purpose of this function is to make it easy for users to create a new class instance with sensible default values. In addition to creating a new class instance, each constructor function does the following:

1. Calls `ownerToMatrix` twice. This method transforms the pre- and post-merger ownership information into a matrix of 1s and 0s if the ownership information is not already in that format.

2. Calls `calcSlopes`. This method calibrates the demand parameters associated with a particular demand system.

3. Calls `calcMC` twice. The first call computes pre-merger marginal costs and the second call calculates post-merger marginal costs, which equals pre-merger marginal costs multiplied by the user-supplied proportional change in marginal costs (1 + 'mcDelta'). The results from this call are assigned to the appropriate class slot,

4. Returns the class instance.

5. Calls `calcPrices` twice. The first call computes pre-merger equilibrium prices and the second call calculates post-merger equilibrium prices. The results from this call are assigned to the appropriate class slot,

6. Returns the class instance.

Perhaps the easiest way to modify an existing class is to create a new child class of that class. That child will inherit all of the parent classes slots and methods. Additional slots may then be easily added and the behavior of existing methods may then be overridden.

7.2 The Auction Models

Currently, the two auction classes available in `antitrust` are *Auction2ndLogit* and *Auction2ndCap*. *Auction2ndLogit* is a child class of *Logit* and therefore follows the same pattern as the Bertrand Model described above.

In contrast, the constructor function for *Auction2ndLogit* is `auction2nd.cap`, which initializes a class instance with sensible default values and then does the following:

1. Calls `calcSellerCostParms`. This method calibrates the the parameters of the seller cost distribution.

2. Calls `calcBuyerValuation`. This method calibrates the cost to the buyer of self supply.

3. Calls `calcOptimalReserve`. Computes the buyer's optimal pre-merger reserve price. If 'constrain.reserve' equals TRUE (the default) the post-merger reserve is set equal to the pre-merger reserve. If 'constrain.reserve' equals FALSE, the optimal post-merger reserve is calculated. The results from this call are assigned to the appropriate class slot.

4. Calls `calcPrices` twice. The first call computes pre-merger equilibrium prices and the second call calculates post-merger equilibrium prices. The results from this call are assigned to the appropriate class slot.

5. Calls `calcMC` twice. The first call computes pre-merger equilibrium marginal costs and the second call calculates post-merger equilibrium marginal costs. The results from this call are assigned to the appropriate class slot.

6. Returns the class instance.

Table 7.1: `antitrust` functions and their information requirements

Name	Model	Price	Margin	Diversion	Quantity/ Share	Capacity	Cite
	Bertrand						
`linear`	Linear	A	1+	O	A		?
`loglin`	Log-linear	A	A	O	A		?
`aids`	AIDS	O	2+	O	A		?
`pcaids`	PCAIDS	O	1	O	A		?
`pcaids.nests`	Nested PCAIDS	O	2+	O	A		?
`logit`	Logit	A	1+		A		?
`logit.alm`	Logit – Unobserved Outside Share	A	2+		A		?
`logit.nests`	Nested Logit	A	2+		A		?
`logit.cap`	Capacity-Constrained Logit	A	2+		A	A	?
`ces`	Constant Elasticity	A	1+		A		?
`ces.nests`	Nested Constant Elasticity	A	2+		A		?
`sim`		A					
	Cournot						
`cournot`	Linear, Log-Linear	A	1+		A		?
	Auctions						
`auction2nd.logit`	2nd price, Logit	O	1+		A		?
`auction2nd.cap`	2nd price, homoegeneous single unit	1+	1+		A	A	?
	Other Tools						
`cmcr.bertrand`	any	M	M	M			?
`cmcr.cournot`	any		1	M	M		?
`upp.bertrand`	any	M	M	M			?
`HHI`	any				A		?

'M': data on merging parties' products,
'A': data on all products,
'O': optional data; if supplied it must be on all products,
'#+': data on at least # products.

Name	Description
CV	Compute compensating variation
calcMC	Compute pre- and post-merger (constant) marginal costs
calcMargins	Compute pre- and post-merger equilibrium margins
calcPrices	Compute pre- and post-merger equilibrium prices
calcPriceDelta	Compute *proportional* change in equilibrium prices
calcProducerSurplus	Compute pre- and post-merger producer surplus
calcProducerSurplusGrimTrigger	Compare pre- and post-merger surplus from colluding to surplus from defecti
calcShares	Compute pre- and post-merger equilibrium shares
cmcr	Compute compensating marginal cost reduction (CMCR)
HypoMonTest	Use the Hypothetical Monopolist Test to
	determine whether a specified set of products satisfy a SSNIP
diversion	Compute pre- and post-merger diversion matrices
diversionHypoMon	Compute the diversion matrix under a Hypothetical Monopolist Test
elast	Compute pre- and post-merger elasticity matrices
hhi	Compute HHI using pre- and post-merger equilibrium shares
upp	Compute net Upwards Pricing Pressure (UPP)
plot	Plot pre- and post-merger demand, marginal costs, and equilibria
summary	Summarize result

Table 7.2: Selected `antitrust` methods. While all of the above methods are defined
for the Bertrand Class and its child classes, not all of these methods are
currently defined for the Auction2ndCap class.

Table 7.3: Compensating variation formulas

Demand	Formula	Reports
Linear	$\alpha'(p_{post} - p_{pre}) + .5 p'_{post} B p_{post} - .5 p'_{pre} B p_{pre}$	level
AIDS	$\alpha'(\log p_{post} - \log p_{pre}) + .5 \log p'_{post} B \log p_{post} - .5 \log p'_{pre} B \log p_{pre}$	proportion
CES	$\dfrac{1}{1+\alpha}\dfrac{1}{1-\gamma} \log\left(\dfrac{\sum\limits_{i \in n} \delta_i p_{i,post}^{1-\gamma}}{\sum\limits_{i \in n} \delta_i p_{i,pre}^{1-\gamma}}\right)$	proportion
Nested CES	$\dfrac{1}{1+\alpha}\dfrac{1}{1-\gamma} \log\left(\dfrac{\sum\limits_{h \in H}\left(\sum\limits_{i \in h} \delta_i p_{i,post}^{1-\sigma_h}\right)^{\frac{1-\gamma}{1-\sigma_h}}}{\sum\limits_{h \in H}\left(\sum\limits_{i \in h} \delta_i p_{i,pre}^{1-\sigma_h}\right)^{\frac{1-\gamma}{1-\sigma_h}}}\right)$	proportion
Logit	$\dfrac{1}{\alpha} \log\left(\dfrac{\sum\limits_{i \in n} \exp(\delta_i + \alpha p_{i,post})}{\sum\limits_{i \in n} \exp(\delta_i + \alpha p_{i,pre})}\right)$	level
Nested Logit	$\dfrac{1}{\alpha} \log\left(\dfrac{\sum\limits_{h \in H}\left(\sum\limits_{i \in h} \exp(\frac{\delta_i + \alpha p_{i,post}}{\sigma_h})\right)^{\sigma_h}}{\sum\limits_{h \in H}\left(\sum\limits_{i \in h} \exp(\frac{\delta_i + \alpha p_{i,pre}}{\sigma_h})\right)^{\sigma_h}}\right)$	level

Note: The 'Reports' column indicates whether the compensating variation formula returns compensating variation in levels (e.g. dollar amounts) or as a proportion of aggregate income.

The Relationship Between Classes in the 'antitrust' Package
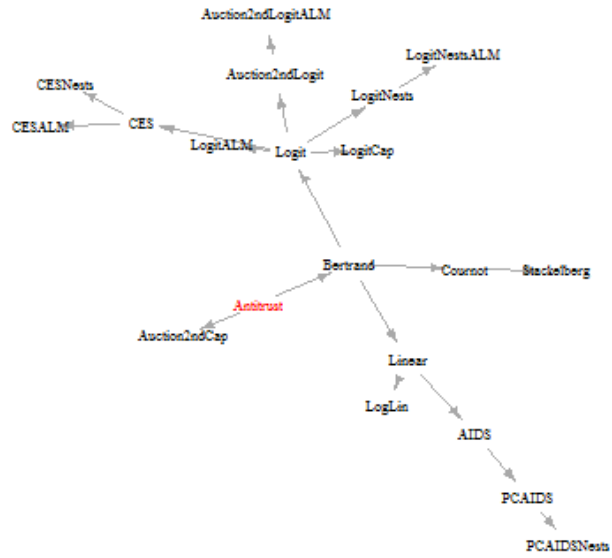
Bibliography

Peter Davis. Coordinated effects merger simulation with linear demands. *Competition Commission*, 2006.

Peter Davis and Cristian Huse. Estimating the 'coordinated effects' of mergers. May 2009.

Roy J. Epstein and Daniel L. Rubinfeld. Merger simulation with brand-level margin data: Extending pcaids with nests. *The B.E. Journal of Economic Analysis & Policy*, advances.4(1):2, 2004. URL `http://econpapers.repec.org/RePEc:bpj:bejeap:v:advances.4:y:2004:i:1:n:2`.

Joseph Farrell and Carl Shapiro. Antitrust evaluation of horizontal mergers: An economic alternative to market definition. *The B.E. Journal of Theoretical Economics*, 10(1):1–39, 2010. doi: 10.2202/1935-1704.1563.

Luke Froeb, Steven Tschantz, and Philip Crooke. Bertrand competition with capacity constraints: mergers among parking lots. *Journal of Econometrics*, 113(1):49 – 67, 2003. ISSN 0304-4076. doi: 10.1016/S0304-4076(02)00166-5. URL `http://www.sciencedirect.com/science/article/pii/S0304407602001665`. <ce:title>Introduction to statistics and econometrics in ligitation support</ce:title>.

Luke M. Froeb and Gregory J. Werden. A robust test for consumer welfare enhancing mergers among sellers of a homogeneous product. *Economics Letters*, 58(3): 367 – 369, 1998. ISSN 0165-1765. doi: DOI:10.1016/S0165-1765(97)00287-5. URL `http://www.sciencedirect.com/science/article/B6V84-3T51RH8-3Y/2/6610d482809009001e3c4e8533d32644`.

Sonia Jaffe and Eric G. Weyl. The first-order approach to merger analysis. *SSRN eLibrary*, 2012.

Jeffrey T. LaFrance. The structure of constant elasticity demand models. *American Journal of Agricultural Economics*, 68(3):543–552, 1986.

Jeffrey T. LaFrance. Integrability of the linear approximate almost ideal demand system. *Economics Letters*, 84(3):297 – 303, 2004. ISSN 0165-1765. doi: 10.1016/j.

econlet.2003.12.019. URL http://www.sciencedirect.com/science/article/pii/S0165176504000461.

Nathan H Miller. Modeling the effects of mergers in procurement. *International Journal of Industrial Organization*, 37:201–208, 2014.

Steven Salop and Daniel O'Brien. Competitive effects of partial ownership: Financial interest and corporate control'. *Antitrust L.J.*, 67:559–614, 2000.

Gloria Sheu. Price, quality, and variety: Measuring the gains from trade in differentiated products. January 2011.

U.S Department of Justice and the Federal Trade Commission. Horizontal merger guidelines. Technical report, 2010.

Roger H. von Haefen. A complete characterization of the linear, log-linear, and semi-log incomplete demand system models. *Journal of Agricultural and Resource Economics*, 27(02), December 2002. URL http://ideas.repec.org/a/ags/jlaare/31118.html.

Keith Waehrer and Martin K. Perry. The effects of mergers in open-auction markets. *The RAND Journal of Economics*, 34(2):287–304, 2003. ISSN 0741-6261.

Gregory J. Werden. A robust test for consumer welfare enhancing mergers among sellers of differentiated products. *The Journal of Industrial Economics*, 44(4):pp. 409–413, 1996. ISSN 00221821. URL http://www.jstor.org/stable/2950522.

Gregory J Werden and Luke M Froeb. The effects of mergers in differentiated products industries: Logit demand and merger policy. *Journal of Law, Economics and Organization*, 10(2):407–26, October 1994. URL http://ideas.repec.org/a/oup/jleorg/v10y1994i2p407-26.html.

Robert D. Willig. Consumer's surplus without apology. *The American Economic Review*, 66(4):pp. 589–597, 1976. ISSN 00028282. URL http://www.jstor.org/stable/1806699.