The `antitrust` Package

Charles Taragin* Michael Sandfort*

October 23, 2012

Contents

*The views expressed herein are entirely those of the authors and should not be purported to reflect those of the U.S. Department of Justice. The `antitrust` package has been released into the public domain without warranty of any kind, expressed or implied. We thank Ronald Drennan, Robert Majure, Russell Pittman, Gloria Sheu, Nathan Miller, Marc Remer, Alexander Raskovich, William Drake, Thomas Jeitschko and Conor Ryan. Address: Economic Analysis Group, Antitrust Division, U.S. Department of Justice, 450 5th St. NW, Washington DC 20530. E-mail: charles.taragin@usdoj.gov and michael.sandfort@usdoj.gov.

`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 demand models as well as simulate the effects of a horizontal merger in a number of different strategic environments.[1] . The output generated by these tools typically include 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 three 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 amongst practitioners.

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 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 example, 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 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`.

[1] Currently, most of `antitrust`'s functions assume that firms are playing a Bertrand differentiated product pricing game. Future versions may support the Cournot game as well as auction models.

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 1 contain examples illustrating how to use the function.

1 The Bertrand Pricing Game

Currently, virtually all of the functions included in `antitrust` assume that firms are playing a one-shot, static Bertrand differentiated product pricing game.[2] Suppose that there are K firms in a market, and that each of the $k \in K$ firms produces n_k products.[3] Let $n = \sum_{k=1}^{K} n_k$ 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, and costs.

Functions in `antitrust` also adopt the additional assumption that 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 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:

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

where 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):

[2] The only exception is `cmcr.cournot` which assumes that firms are playing a Cournot quantity-setting game.

[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.

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

which may be rewritten as

$$FOC_i \equiv r_i + \sum_{j=1}^{n_k} r_j m_j \epsilon_{ji} = 0 \qquad \text{for all } i \in n_k,$$

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:

$$r + (E \circ \Omega)'(r \circ m) = 0 \tag{1}$$

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 Ω is an $n \times n$ matrix whose i, jth element equals the share of product j owned by the firm setting product i's price.[4] In many cases, products 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. '$\circ$' is the Hadamard (entry-wise) product operator.

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.[5]

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 $\{g_i\}_{i=1}^{n_k}$. Mathematically, firm k solves:

[4] 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.

[5] This section is based on the model described in Froeb et al. [2003, p. 51-55]

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

subject to

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

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 $FOC_i \leq 0$ and $q_i - g_i = 0$, while in the latter $FOC_i = 0$ and $q_i - g_i \leq 0$. Mathematically, these cases can be written as

$$\max\{FOC_i, q_i - g_i\} = 0, i = 1 \ldots n_k \tag{2}$$

2 Calibrating Model Demand and Cost Parameters

Although most of the functions listed in Table 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 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 Table 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. For capacity-constrained models, equation 2 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

$$\begin{aligned} d_{ij}^q &= -\frac{\frac{\partial q_j}{\partial p_i}}{\frac{\partial q_i}{\partial p_i}} \\ &= -\frac{\epsilon_{ji} q_j}{\epsilon_{ii} q_i} \end{aligned} \tag{3}$$

$$\begin{aligned} d_{ij}^r &= -\frac{\frac{\partial p_j q_j}{\partial p_i}}{\frac{\partial p_i q_i}{\partial p_i}} \\ &= -\frac{\epsilon_{ji}(\epsilon_{jj}-1) r_j}{\epsilon_{jj}(\epsilon_{ii}-1) r_i} \end{aligned} \tag{4}$$

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.

Although diversion ratios are not present in either equation 1 or 2, 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 or 2, 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 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.

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

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

which may be written in matrix notation as

$$q = \alpha + Bp,$$

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 the following own- and cross-price elasticities:

$$\epsilon_{ii} = \beta_{ii} \frac{p_i}{q_i}$$
$$\epsilon_{ij} = \beta_{ij} \frac{p_i}{q_j}$$

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.[6] With the linear model, this assumption reduces the number of unknown parameters to $2n$, 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 and only if B is a symmetric matrix.[7] Imposing this additional assumption reduces the number of unknown parameters to $n + 1$ (n intercepts and 1 slope), which means that the system is over-identified.[8]

By default, the `calcSlopes` method, called by the `linear` function to calibrate the linear demand parameters, assumes that the above two assumptions hold and uses all the FOCs from equation 1 for which there is sufficient information to solve for the $n + 1$ unknown parameters. Under these assumptions, users need only to supply margin information for a single firm's products to uncover the unknown slope parameter. However, `calcSlopes` allows for the possibility that users may possess margin information on multiple products, in which case there are more equations than unknowns. To accommodate this, `calcSlopes` employs a minimum distance algorithm to find the parameter value that best satisfies all the FOCs for which there are data.

For completeness, `linear` includes the 'symmetry' argument that, when set equal to FALSE, instructs `calcSlopes` to calibrate demand parameters without imposing symmetry 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]

[6] By default, `linear` assumes diversion according to quantity share. Diversion according to quantity share assumes that $d^q_{ij} = \frac{s_j}{1-s_i}$, 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.

[7] See von Haefen [2002].

[8] Mathematically, symmetry implies that $\beta_{ij} = \beta_{ji}$, for all products i, j. Under linear demand, the definition of quantity diversion can be rewritten as $\beta_{ji} = -d^q_{ij}\beta ii$, Combining these assumptions implies that $\beta_{jj} = \frac{d^q_{ij}}{d^q_{ji}}\beta_{ii}$. Hence, knowing a single β_{ii} for some $i \in n$ is sufficient to estimate all the diagonal elements of B, which in turn are sufficient to estimate all the off-diagonal elements of B. Once B has been estimated, α may be recovered from the linear demand system.

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

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 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.

[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 LaFrance [1986] and von Haefen [2002] for more details.

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

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

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:

$$\epsilon_{ii} = -1 + \frac{\beta_{ii}}{r_i} + r_i(1 + \epsilon)$$
$$\epsilon_{ij} = \frac{\beta_{ij}}{r_i} + r_j(1 + \epsilon),$$

where ϵ is the market elasticity of demand. Some implications of the LA-AIDS are that $\epsilon \leq -1$ and $|\epsilon| \leq |\epsilon_{ii}|$ for all $i \in n$.

As with the linear demand system, the LA-AIDS model assumes that B is symmetric 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 $n + 1$ unknown demand parameters (n intercepts and 1 slope) and up to $2n$ equations, in which case the system is over-identified. Therefore, only the margin information for a single firm's products is needed to estimate all the price coefficients.

[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 derive the elasticities.

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

One interesting feature of the LA-AIDS that distinguishes it from the other demand systems included in `antitrust` is that LA-AIDS elasticities incorporate ϵ, the market elasticity parameter. 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. 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 FOCs than unknowns to identify both the unknown demand parameter described above as well as ϵ. This means that there are $n + 2$ unknown parameters and therefore users must supply `aids` with at least two product margins in order to uncover estimates for both the unknown demand parameter and ϵ.[13] If either only a single product margin is observed, or ϵ is observed, then `pcaids` may be used in lieu of `aids` to calibrate the LA-AIDS parameters.[14]

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.

2.3.1 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.

[13] ϵ and the unknown demand parameter must also satisfy the inequality restrictions described above as well as the $2n$ FOCs. The `calcSlopes` method uses the `constrOptim` function to find the parameter estimates that satisfy these inequality constraints.

[14] The main difference between `pcaids` and `aids` is that while `aids` requires users to supply revenue shares and at least two margins 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.

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.[15] 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.[16] Accordingly, users must supply margin information for at least $\frac{H(H-1)}{2}$ products.[17]

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

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

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

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

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

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

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).

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

[16] 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.

[17] Note that these margins are in addition to the margin information that may be necessary to identify the elasticity of a single product ('knownElast').

`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.[18] Otherwise, an outside good is included whose price and δ are normalized to 0, and whose share equals $s_0 = 1 - \sum_{i \in n} s_i$.

2.4.1 Logit With Unobserved Outside Share

The Bertrand model with Logit demand and unobserved outside share may be implemented using the `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.[19]

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

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

[18] 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.

[19] 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.

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

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

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

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

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.

2.4.2 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, 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.

To determine whether a capacity constraint is binding pre-merger, Logit quantity shares must be transformed into levels. This is accomplished by noting that $s_i \equiv \frac{q_i}{mktSize}$, and employing the user-supplied value for 'mktSize' to recover q_i.

2.4.3 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.[20] `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

$$\begin{aligned} s_i =& s_{i|h} s_h, \\ s_{i|h} =& \frac{\exp(\frac{V_i}{\sigma_h})}{\sum_{k \in h} \exp(\frac{V_k}{\sigma_h})}, & 1 \geq \sigma_h \geq 0 \\ s_h =& \frac{\exp(\sigma_h I_h)}{\sum_{l \in H} \exp(\sigma_l I_l)}, & I_h = \log \sum_{k \in h} \exp\left(\frac{V_k}{\sigma_h}\right). \end{aligned}$$

We assume that V_i takes on the following form

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

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

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

[20] 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.

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.

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.[21]

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

[21] 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.

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

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

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

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

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

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.[22] Otherwise, an outside good is included whose price and δ are normalized to 1, and whose share equals $r_0 = 1 - \sum_{i \in n} r_i$.

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

[22] 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.

the consumer is willing to spend on the $n+1$ products that are within the market.[23] By default, 'shareInside' equals 1, which indicates that the customer spends her entire budget on the $n+1$ products within the market.

2.5.1 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.[24] 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.[25] `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

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

We assume that V_i takes on the following form

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

where $\sigma_h > \gamma > 1$ for all nests $h \in H$. The Nested Logit demand system yields the

[23] 1-'shareInside' equals the proportion of the representative consumer's income that is spent on all other products (i.e. the numeraire).

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

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

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.

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.[26] 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.[27] 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.[28] 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.

3 Simulating Merger Effects

For most of the demand systems included in `antitrust`, a closed-form solution in prices to the FOCs equation does not exist. We therefore employ the non-linear equation solver in the `nleqslv` package to find equilibrium prices. It is worth noting that the FOCs in equation 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.[29] Practitioners sometimes address this problem by starting the non-linear

[26] 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 $m = \hat{m}_{pre}$.

[27] 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%.

[28] To see why, note that equation 2 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.

[29] 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

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 `nleqslv` used by most `antitrust` functions solves for the pre- and post-merger price equilibrium. Modifications to `nleqslv`'s default behavior may be accomplished by including `nleqslv` arguments in any of the `antitrust` functions described above.[30] See `nleqslv`'s help page for more information on how to modify `nleqslv`'s behavior.

The FOCs for the capacity-constrained Bertrand game (equation 2) 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. Froeb et al. [2003, p. 54] suggests replacing equation 2 with

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

which has the same roots as equation 2, 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` contains methods that can compute many other features of the model. Table 2 lists some of the methods that users may find interesting.

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.[31] *Quantity* shares rather than

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

[30] `linear`'s `calcPrices` method employs `constrOptim` rather than `nleqslv`.

[31] 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 only if the user supplied pre-merger prices. Compensating variation is only reported for the Linear model if the matrix of slope coefficients is symmetric.

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.

3.2 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.[32]

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.

3.3 Measuring Changes In Consumer Welfare

All of the demand models included in `antitrust` 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 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

[32]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.

levels (e.g. dollars) or as a percent of the representative consumer's total income.[33]

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 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.

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.[34]

3.4 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. [U.S Department of Justice and the Federal Trade Commission, 2010, p. 7]

To assist users in identifying antitrust product and geographic markets, `antitrust` includes the `HypoMonTest` method. `HypoMonTest` assumes that i) firms are playing

[33]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.

[34]See Willig [1976].

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. [35]

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. 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 [U.S Department of Justice and the Federal Trade Commission, 2010, p. 9].

To facilitate such comparisions, `antitrust` includes the `diversionHypoMon` method, which, for a set of products specified using the 'prodIndex' argument, returns the revenue diversion (as defined by equation 4)[36] 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).

[35] 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. [U.S Department of Justice and the Federal Trade Commission, 2010, p. 9]

The Guidelines describe a similar test for geographic market definition [U.S Department of Justice and the Federal Trade Commission, 2010, p. 13]

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

3.5 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. To accommodate this, `antitrust` contains the `sim` function.The `sim` function allows users to simulate price effects (or the output from any method listed in Table 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.

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

$$\begin{aligned} m_{post} =& (B_{post})^{-1} B_{pre} m_{pre} \\ B_s =& D^q_{pre} \circ ((1/p_{pre}) p'_{pre}) \circ \Omega_s, \qquad\qquad s \in \{pre, post\}, \end{aligned}$$

where D^q_{pre} 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:

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

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 Net UPP

Another measure included in `antitrust` is net Upward Pricing Pressure (UPP). Net UPP 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

net UPP for all of firm k's products following a merger with firm l may be written as

$$
\begin{aligned}
UPP_k =& B_{kk,pre}^{-1} B_{kl,post}(m_l \circ p_l) - (1 - m_k) \circ p_k \circ \Delta mc_k \\
B_{fg,s} =& D_{fg,pre}^{q} \circ ((1/p_{f,pre}) p'_{g,pre}) \circ \Omega_{fg,s}, \qquad\qquad s \in \{pre, post\}; f, g \in \{k, l\}
\end{aligned}
$$

where $D_{fg,pre}^{q}$ is an $n_f \times n_g$ matrix whose f,gth element is the pre-merger quantity diversion from product $f \in n_f$ to product $g \in n_g$, $p_{f,pre}$ and $p_{g,pre}$ are length-n_f and length-n_g vectors of prices, and $\Omega_{fg,s}$ is an $n_f \times n_g$ matrix of either pre- or post-merger ownership shares (typically equal to 1). Δmc_k a length-n_k vector containing the anticipated proportional changes in the marginal costs of firm k's products due to the merger. Net UPP predicts that the acquiring firm will have an incentive to raise the price of product $i \in n_k$ when $UPP_k > 0$.[37]

`upp.bertrand` may be used to compute UPP 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." [U.S Department of Justice and the Federal Trade Commission, 2010, 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

[37] Farrell and Shapiro [2010] introduce net UPP. Jaffe and Weyl [2012] demonstrate how net UPP can be extended to accommodate multi-product merging firms.

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.

5 Gotchas

While in many instances, `antitrust` functions produce reasonable predictions, we have noticed that at times `antitrust` functions will either fail to produce a result, or produces results that are not intuitive. Here, we highlight some instances where `antitrust` produces seemingly unintuitive results, and explain the likely cause of this behavior.

5.1 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.

5.2 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.

5.3 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 combina-

tions 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 problem appears to occur because the set of FOCs that are being used to calibrate this parameter are nearly co-linear. This co-linearity problem 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.

6 Modifying and Extending `antitrust`

`antitrust` was written using `R`'s S4 object-oriented class system. Figure 1 displays the relationships between each parent-child class. The figure indicates that the *Bertrand* class is the main class. Indeed, every effort has been made to include in the *Bertrand* class all slots that are common to its child classes as well as all common methods.

Figure 1 also reveals that each of *Bertrand*'s child classes is 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 `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,
4. 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.

Table 1: `antitrust` functions and their information requirements

Name	Demand	Price	Margin	Diversion	Quantity/ Share	Capacity	Cite
`cmcr.bertrand`	any	M	M	M			Werden [1996]
`cmcr.cournot`	any		1		M		Froeb and Werden [1998]
`upp.bertrand`	any	M	M	M			Farrell and Shapiro [2010] Jaffe and Weyl [2012]
`HHI`	any				A		Salop and O'Brien [2000]
`linear`	Linear	A	1+	O	A		von Haefen [2002]
`loglin`	Log-linear	A	A	O	A		von Haefen [2002] LaFrance [2004]
`aids`	AIDS	O	2+	O	A		Epstein and Rubinfeld [2004] LaFrance [2004]
`pcaids`	PCAIDS	O	1	O	A		Epstein and Rubinfeld [2004] LaFrance [2004]
`pcaids.nests`	Nested PCAIDS	O	2+	O	A		Epstein and Rubinfeld [2004]
`logit`	Logit	A	1+		A		Werden and Froeb [1994]
`logit.alm`	Logit – Unobserved Outside Share	A	2+		A		Werden and Froeb [1994]
`logit.nests`	Nested Logit	A	2+		A		Werden and Froeb [1994]
`logit.cap`	Capacity-Constrained Logit	A	2+		A	A	Froeb et al. [2003]
`ces`	Constant Elasticity	A	1+		A		Sheu [2011]
`ces.nests`	Nested Constant Elasticity	A	2+		A		Sheu [2011]
`sim`		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.

Table 2: Selected `antitrust` methods

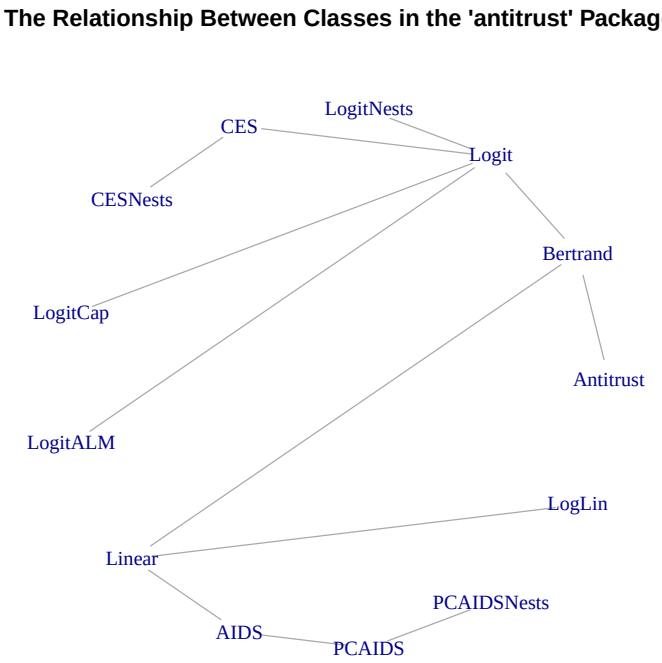
Name	Description
`CV`	Compute compensating variation
`calcMC`	Compute pre- and post-merger (constant) marginal costs
`calcMargins`	Compute pre- and post-merger equilibrium margins
`calcPrices`	Compute pre- and post-merger equilibrium prices
`calcPriceDelta`	Compute *proportional* change in equilibrium prices
`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 equilbrium shares
`upp`	Compute net Upwards Pricing Pressure (UPP)
`summary`	Summarize result

Table 3: Compensating variation formulas

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

Figure 1:



References

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.

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.

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.