Package 'healthcare.antitrust'

July 19, 2026

Type Package

Title Healthcare Antitrust Analysis

Version 0.1.6

Description Antitrust analysis of
healthcare markets. Contains functions to implement the
semiparametric estimation technique described in Raval, Rosenbaum,
and Tenn (2017) ``A Semiparametric Discrete Choice Model: An Application
to Hospital Mergers" <doi:10.1111/ecin.12454>.

License CC0

Copyright As a work of the United States government, this project is
in the public domain within the United States. Additionally, we
waive copyright and related rights in the work worldwide
through the CC0 1.0 Universal public domain dedication.

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

Imports methods, stats

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.2

URL https://github.com/mpanhans/healthcare.antitrust

NeedsCompilation no

Author Matthew T Panhans [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7874-6740>)

Maintainer Matthew T Panhans <mpanhans@gmail.com>

Repository CRAN

Date/Publication 2026-07-19 19:40:02 UTC

Contents

cell_defn 2
discharge_data 3
div_calc 4
impliedshare_calc 5
wtp_calc 7

Index **9**

`cell_defn` *Allocate Observations to Cells*

Description

Take a dataset of hospital discharges, and assigns them to cells that meet a minimum threshold cell size.

Usage

```
cell_defn(data, min_size, layers, count = "count", expandLayers = FALSE)
```

Arguments

`data`	Dataset of discharges.
`min_size`	Minimize cell size.
`layers`	A list of lists. Each layer is a list of the variables on which observations will be allocated to cells. The layers should be ordered in decreasing refinement, so that observations not assigned to a cell meeting the minimum size threshold can be assigned by a more coarse layer.
`count`	Name of variable which indicates the number of admissions represented by an observation. Default variable name is `count`.
`expandLayers`	logical; if only a single layer is given, setting this to TRUE will expand the layer to a set of layers where each layer drops the last variable in the list of the previous layer.

Details

If the variable `count` is not available, the function will assume that each observation represents one admission, and a variable `count` will be created in the output data frame to indicate this.

When assigning observations to cells in a given layer, this function only assigns observations that have not been previously assigned in a finer layer. That is, the function assigns observations without replacement. Assignment with replacement is currently not supported.

For more details see the example vignette by typing: `vignette("semipar_example", package = "healthcare.antitrust")`

Value

A list of data frames. The first component in the list, `assigned`, is the original data frame, with observations assigned to cells, and excluding observations that were not assigned. The assigned cells are designated by the variable `cell`, and the layer number in which the observation is assigned to the cell is given by the variable `cell_type`. The second component of the list, `unassigned`, is a data frame with the unassigned observations.

Examples

```
data(discharge_data, package = "healthcare.antitrust")

list1 <- c("drg","age","zip5")
layers <- list(list1)
th <- 15
discharge_data$count <- 1

outList <- cell_defn(discharge_data,th,layers)
```

---

discharge_data *A simulated dataset of hospital discharges*

Description

A simulated dataset of hospital discharges containing patient characteristics.

Usage

```
discharge_data
```

Format

A data frame with 1,200 rows (one for each discharge) and 7 variables:

drg diagnosis related group (drg) code for the admission

age patient age

zip5 patient 5 digit zip code of residence

hosp_id hospital identifier

hospital hospital name

sys_id system identifier

system system name

Source

Created as part of healthcare.antitrust package.

`div_calc` *Diversion Ratio Calculator*

Description

Calculates provider-level diversion ratios, once cells have been defined.

Usage

```
div_calc(
  data,
  cell = "cell",
  provider_id = "provider_id",
  provider = "provider",
  sys_id = "sys_id",
  focal_sys_id = NULL,
  party_ind = "party_ind",
  count = "count",
  dropDegenerateCell = TRUE,
  hosp_id = NA,
  hospital = NA
)
```

Arguments

`data`	Dataset of patient choices, with required variables: `cell`, `provider_id`, `hospital`, `sys_id`, `party_ind`, `count`. Use other function arguments to indicate alternative variable names to the default names.
`cell`	Name of variable specifying cell to which each observation has been allocated. Default variable name is `cell`. Can be created by `cell_defn` function.
`provider_id`	Name of variable specifying (numeric) provider identifier. Default variable name is `provider_id`.
`provider`	Name of variable specifying (string) provider name. Default variable name is `provider`.
`sys_id`	Name of variable specifying (numeric) system identifier. Default variable name is `sys_id`.
`focal_sys_id`	numeric; list of sys_id's specifying systems of interest for which diversions will be calculated from. For a merger, this will typically be the system identifies of the merging parties.
`party_ind`	deprecated; use focal_sys_id instead.
`count`	Name of variable indicating the number of admissions represented by the observation. Set = 1 for every row if each observation represents one admission.
`dropDegenerateCell`	logical; specifies how to treat cells with a 100 percent within-system share. If TRUE, observations in degenerate, 100 percent share cells will be ignored in the

	diversion ratio calculation. If FALSE, any such individuals will be assigned to the outside option, but still included in the denominator, so that the inside-option diversion will total less than 100 percent.
hosp_id	deprecated; use provider_id instead.
hospital	deprecated; use provider instead.

Details

For system-to-system diversions, set provider_id and provider equal to corresponding system-level identifiers. Diversions then reflect that patients are not allowed to divert to within-system alternative providers

For more details see the example vignette by typing: vignette("semipar_example", package = "healthcare.antitrust")

Value

A list with two components. The first component, provider_level, is a matrix giving provider-level diversions from party providers to all other providers The second object, sys_level, is a matrix that aggregates party providers to systems, thus giving diversions from party systems to all other providers

Examples

```
data(discharge_data, package = "healthcare.antitrust")

list1 <- c("drg","age","zip5")
layers <- list(list1)
th <- 15
discharge_data$count <- 1

outList <- cell_defn(discharge_data,th,layers)
D0 <- outList$assigned

out <- div_calc(D0, provider_id = "hosp_id", provider = "hospital",
          focal_sys_id = c(1,5))
```

---

impliedshare_calc *Implied Market Share Calculator*

Description

Calculate implied market shares based on diversion ratios.

Usage

```
impliedshare_calc(
  M,
  div_from,
  quantities,
  focal,
  div_to,
  reference_goods,
  returnM = FALSE
)
```

Arguments

M	Market size.
div_from	Vector of diversions from the focal good to other goods in the market, with each element between 0 and 1.
quantities	Vector of quantities or revenues for each good in the market.
focal	Scalar indicating the index of the focal good.
div_to	Vector of diversions to the focal good from other goods in the market, with each element between 0 and 1.
reference_goods	Vector indicating non-focal good(s) with diversion to focal good known, used to determine the market size.
returnM	logical; default to FALSE, in which case function returns implied shares for each good in the market and corresponding membership weight for each good. If TRUE, function can be used to find value of M which best matches observed diversions to focal good.

Details

This function calculates implied market shares based on given diversion ratios.

For more details see the example vignette by typing: `vignette("semipar_example", package = "healthcare.antitrust")`

Value

A list with two components. The first component, `s_implied`, is a vector of implied market shares for each good in the market. The second object, `weights`, is a vector of each good's corresponding membership weight in the market.

Examples

```
q_1 <- 40
q_2 <- 20
div_1_2 <- .3333
div_2_1 <- .50
```

```
find_M <- optimize(f = impliedshare_calc, lower = 50, upper = 250,
   div_from=c(0,div_1_2), quantities=c(q_1,q_2), focal=1,
   div_to=c(0,div_2_1), reference_goods = c(2), returnM = TRUE)

find_M
M_implied <- find_M$minimum


impliedshare_calc(M=M_implied,div_from=c(0,div_1_2),
   quantities=c(q_1,q_2),
   focal=1, div_to=c(0,div_2_1),reference_goods = c(2))
```

`wtp_calc` *Willingness-to-Pay Calculator*

Description

This function calculates the system-level Willingness-To-Pay.

Usage

```
wtp_calc(
  data,
  cell = "cell",
  sys_id = "sys_id",
  count = "count",
  weight = "weight",
  dropDegenerateCell = TRUE
)
```

Arguments

`data`	Dataset of hospital discharges. Required variables: `cell` and `sys_id`. Use other function arguments to indicate alternative variable names to the default names.
`cell`	Name of variable specifying cell to which each observation has been allocated. Default variable name is `cell`. Can be created by `cell_defn` function.
`sys_id`	Name of variable specifying (numeric) system identifier. Default variable name is `sys_id`.
`count`	Name of variable indicating the number of admissions represented by the observation. Set = 1 for every row if each observation represents one admission.
`weight`	Name of variable indicating designated weight of admission; =1 if observations should be equally weighted.
`dropDegenerateCell`	logical; specifies how to treat cells with a 100 percent within-system share. If TRUE, observations in degenerate, 100 percent share cells will be ignored in the WTP calculation. If FALSE, an adjustment is made where any cells with > 99 percent share at a single hospital have the share set to 99.0 percent.

Details

This function calculates the system-level WTP. For use in a WTP simulation exercise of the "WTP/Q Method" described in Brand and Balan (2018) doi:10.2139/SSRN.3153109. Hospital systems need to be numbered by `sys_id`, with a distinct `sys_id` for each independent hospital.

The `weight` input might for example be a DRG weight for each inpatient hospital admission.

For more details see the example vignette by typing: `vignette("semipar_example", package = "healthcare.antitrust")`

Value

A data frame with the number of rows equal to the number of systems in the input data frame, and the following variables: `sys_id` is the system identifier, `WTP_s` is the system's WTP, `N_s` is the number of admission for the system, and `WTP_s_wt` is the weighted WTP for the system if the weight input is provided.

Examples

```
data(discharge_data, package = "healthcare.antitrust")

list1 <- c("drg","age","zip5")
layers <- list(list1)
th <- 15
discharge_data$count <- 1

outList <- cell_defn(discharge_data,th,layers)
D0 <- outList$assigned

out <- wtp_calc(D0)
```

Index

∗ **datasets**
discharge_data, 3

cell_defn, 2

discharge_data, 3
div_calc, 4

impliedshare_calc, 5

wtp_calc, 7