Package ‘matchingMarkets’

February 23, 2018

Version 0.3-5
Depends R (>= 3.0.2)
Imports Rcpp (>= 0.12.12), RcppProgress (>= 0.2), rJava, lpSolve, lattice, partitions, parallel, stats, utils
SystemRequirements Java, C++11
LinkingTo Rcpp, RcppProgress, RcppArmadillo
Suggests knitr, ggplot2, grDevices, graphics
VignetteBuilder knitr
Title Analysis of Stable Matchings
Author Thilo Klein [aut, cre], Sven Giegerich [ctb]
Maintainer Thilo Klein <thilo@klein.uk>
Description Implements structural estimators to correct for the sample selection bias from observed outcomes in matching markets. This includes one-sided matching of agents into groups as well as two-sided matching of students to schools. The package also contains algorithms to find stable matchings in the three most common matching problems: the stable roommates problem, the college admissions problem, and the house allocation problem.


BugReports https://github.com/thiloklein/matchingMarkets/issues
License GPL (>= 2)
RoxygenNote 6.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-02-23 22:36:37 UTC
matchingMarkets-package

An R package for the analysis of stable matchings.

Description

The matchingMarkets package contains R, C++ and Java code for stable matching algorithms and the estimation of structural models that correct for the sample selection bias of observed outcomes in matching markets.

Matching is concerned with who transacts with whom, and how. For example, who works at which job, which students go to which school, who forms a workgroup with whom, and so on.

The empirical analysis of matching markets is naturally subject to sample selection problems. If agents match assortatively on characteristics unobserved to the analyst but correlated with both the exogenous variable and the outcome of interest, regression estimates will generally be biased.

The matchingMarkets package comprises

1. Bayes estimators. The estimators implemented in function stabit and stabit2 correct for the selection bias from endogenous matching.

   The current package version provides solutions for two commonly observed matching processes: (i) the group formation problem with fixed group sizes and (ii) the college admissions problem. These processes determine which matches are observed – and which are not – and this is a sample selection problem.

2. Post-estimation tools. Setting mfx=TRUE in the summary function computes marginal effects from coefficients in binary outcome and selection equations and khb implements the Karlson-Holm-Breen test for confounding due to sample selection.
3. **Design matrix generation.** The estimators are based on independent variables for all feasible, i.e., observed and counterfactual, matches in the market. Generating the characteristics of all feasible matches from individual-level data is a combinatorial problem. The package returns design matrices based on pre-specified transformations to generate counterfactual matches.

4. **Algorithms.** The package also contains three matching algorithms that can be used to simulated matching data: **hri**: A constraint model (Posser, 2014) for the stable marriage and college admissions problem, a.k.a. hospital/residents problem (see Gale and Shapley, 1962). **sri**: A constraint model for the stable roommates problem (see Gusfield and Irving, 1989). **ttc**: The top-trading-cycles algorithm for the housing market problem. These can be used to obtain stable matchings from simulated or real preference data (see Shapley and Scarf, 1974).

5. **Data.** In addition to the baac00 dataset from borrowing groups in Thailand’s largest agricultural lending program, the package provides functions stabsim and stabsim2 to simulate one’s own data from one-sided and two-sided matching markets.

**Frequently Asked Questions**

- **Why can I not use the classic Heckman correction?**
  Estimators such as the Heckman (1979) correction (in package sampleSelection) or double selection models are inappropriate for this class of selection problems. To see this, note that a simple first stage discrete choice model assumes that an observed match reveals match partners’ preferences over each other. In a matching market, however, agents can only choose from the set of partners who would be willing to form a match with them and we do not observe the players’ relevant choice sets.

- **Do I need an instrumental variable to estimate the model?**
  Short answer: No. Long answer: The characteristics of other agents in the market serve as the source of exogenous variation necessary to identify the model. The identifying exclusion restriction is that characteristics of all agents in the market affect the matching, i.e., who matches with whom, but it is only the characteristics of the match partners that affect the outcome of a particular match once it is formed. No additional instruments are required for identification (Sorensen, 2007).

- **What are the main assumptions underlying the estimator?**
  The approach has certain limitations rooted in its restrictive economic assumptions.
  1. The matching models are complete information models. That is, agents are assumed to have a complete knowledge of the qualities of other market participants.
  2. The models are static equilibrium models. This implies that (i) the observed matching must be an equilibrium, i.e., no two agents would prefer to leave their current partners in order to form a new match (definition of pairwise stability), and (ii) the equilibrium must be unique for the likelihood function of the model to be well defined (Bresnahan and Reiss, 1991).
  3. Uniqueness results can be obtained in two ways. First, as is common in the industrial organization literature, by imposing suitable preference restrictions. A suitable restriction on agents’ preferences that guarantees a unique equilibrium is alignment (Pycia, 2012). In a group formation model, (pairwise) preference alignment states that any two agents who belong to the same groups must prefer the same group over the other. Second, by choosing a market assignment based on matching algorithms that produce a unique stable matching, such as the well-studied Gale and Shapley (1962) deferred acceptance algorithm.
4. Finally, the models assume *bivariate normality* of the errors in selection and outcome equation. If that assumption fails, the estimator is generally inconsistent and can provide misleading inference in small samples.

**How to cite this package**

Whenever using this package, please cite as

**Author(s)**

Thilo Klein

**References**


**See Also**

sampleSelection

---

**Description**

The baac00 data frame contains data of 292 borrowers from Thailand’s largest agricultural lending program. These data are collected as part of the Townsend Thai Project Bank for Agriculture and Agricultural Cooperatives (BAAC) Annual Resurvey (Townsend, 2000). The 292 borrowers are nested within 68 groups and 39 markets. This nestedness makes the dataset particularly relevant for matching applications. A more complete discussion of the data is found in Ahlin (2009), Section 3, and Klein (2015a).
Usage

data(baac00)

Format

This data frame contains the following columns:

- **g.id**  group identifier.
- **m.id**  market identifier.
- **R**  repayment outcome: BAAC never raised interest rate as a penalty for late repayment.
- **pi**  success probability: measure of group members’ project success probability.
- **wst**  worst year: indicator of economically worst year. 1: last year; 2: year before last year; 101-168: neither.
- **loan_size**  loan size: average loan size borrowed by the group.
- **loan_size2**  loan size squared.
- **lngroup_agei**  log group age: log of number of years group has existed.

Source


References


---

**hri**  
All stable matchings in the hospital/residents problem with incomplete lists

Description

Finds all stable matchings in either the hospital/residents problem (a.k.a. college admissions problem) or the related stable marriage problem. Dependent on the problem, the results comprise the student and college-optimal or the men and women-optimal matchings. The implementation allows for incomplete preference lists (some agents find certain agents unacceptable) and unbalanced instances (unequal number of agents on both sides). The function uses the Prosser (2014) constraint encoding based on either given or randomly generated preferences.
Usage

\texttt{hri(nStudents = ncol(s.prefs), nColleges = ncol(c.prefs), nSlots = rep(1, nColleges), s.prefs = NULL, c.prefs = NULL, seed = NULL, s.range = NULL, c.range = NULL, ...)}

Arguments

\texttt{nStudents} integer indicating the number of students (in the college admissions problem) or men (in the stable marriage problem) in the market. Defaults to \texttt{ncol(s.prefs)}.

\texttt{nColleges} integer indicating the number of colleges (in the college admissions problem) or women (in the stable marriage problem) in the market. Defaults to \texttt{ncol(c.prefs)}.

\texttt{nSlots} vector of length \texttt{nColleges} indicating the number of places (i.e. quota) of each college. Defaults to \texttt{rep(1, nColleges)} for the marriage problem.

\texttt{s.prefs} matrix of dimension \texttt{nColleges} x \texttt{nStudents} with the \texttt{j}th column containing student \texttt{j}'s ranking over colleges in decreasing order of preference (i.e. most preferred first).

\texttt{c.prefs} matrix of dimension \texttt{nStudents} x \texttt{nColleges} with the \texttt{i}th column containing college \texttt{i}'s ranking over students in decreasing order of preference (i.e. most preferred first).

\texttt{seed} integer setting the state for random number generation.

\texttt{s.range} range of two integers \texttt{s.range = c(s.min, s.max)}, where \texttt{s.min} < \texttt{s.max}. Produces incomplete preference lists with the length of each student's list randomly sampled from the range \texttt{[s.min, s.max]}. Note: interval is only correct if either \texttt{c.range} or \texttt{s.range} is used.

\texttt{c.range} range of two integers \texttt{c.range = c(c.min, c.max)}, where \texttt{c.min} < \texttt{c.max}. Produces incomplete preference lists with the length of each college's list randomly sampled from the range \texttt{[c.min, c.max]}. Note: interval is only correct if either \texttt{c.range} or \texttt{s.range} is used.

Value

\texttt{hri} returns a list of the following elements.

\texttt{s.prefs.sm} student-side preference matrix for the stable marriage problem with incomplete lists (SMI).

\texttt{c.prefs.sm} college-side preference matrix for the stable marriage problem with incomplete lists (SMI).

\texttt{s.prefs.hri} student-side preference matrix for the college admissions problem (a.k.a. hospital/residents problem) with incomplete lists (HRI).

\texttt{c.prefs.hri} college-side preference matrix for the college admissions problem (a.k.a. hospital/residents problem) with incomplete lists (HRI).

\texttt{matchings} edgelist of matched students and colleges, including the number of the match (matching) and two variables that indicate the student-optimal match (sOptimal) and college-optimal match (cOptimal).
Minimum required arguments

`hri` requires the following combination of arguments, subject to the matching problem.

- `nStudents`, `nColleges` Marriage problem with random preferences.
- `s.prefs`, `c.prefs` Marriage problem with given preferences.
- `nStudents`, `nSlots` College admissions problem with random preferences.
- `s.prefs`, `c.prefs`, `nSlots` College admissions problem with given preferences.

Author(s)

Thilo Klein

References


Examples

```r
## -----------------------------
## --- Marriage problem

## 3 men, 2 women, random preferences:
  hri(nStudents=7, nColleges=6, seed=4)

## 3 men, 2 women, given preferences:
  s.prefs <- matrix(c(1, 2, 1, 2, 1, 2), 2, 3)
  c.prefs <- matrix(c(1, 2, 2, 3), 3, 2)
  hri(s.prefs=s.prefs, c.prefs=c.prefs)

## -----------------------------
## --- College admission problem

## 7 students, 2 colleges with 3 slots each, random preferences:
  hri(nStudents=7, nSlots=c(3, 3), seed=21)

## 7 students, 2 colleges with 3 slots each, given preferences:
  s.prefs <- matrix(c(1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 2, 7)
  c.prefs <- matrix(c(1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, NA, NA), 7, 2)
  hri(s.prefs=s.prefs, c.prefs=c.prefs, nSlots=c(3, 3))

## 7 students, 3 colleges with 3 slots each, incomplete preferences:
  hri(nStudents=7, nSlots=c(3,3,3), seed=21, s.range=c(1,3))

## -----------------------------
## --- Summary plots
```
## Not run:
## 200 students, 200 colleges with 1 slot each
res <- hri(nStudents=200, nColleges=200, seed=12)
plot(res)
plot(res, energy=TRUE)

## End(Not run)

### hri2

**Resident-optimal matching in the hospital/residents problem with couples**

**Description**

Implements the Roth Peranson matching algorithm for the hospital/residents problem with couples as described in Roth and Peranson (1999). The function is based on an adoption of Fahiem Bacchus’s stable-matching-suite.

**Usage**

```r
hri2(nStudents = ncol(s.prefs), nColleges = ncol(c.prefs), nSlots = rep(1, nColleges), nCouples = ncol(co.prefs), s.prefs = NULL, c.prefs = NULL, co.prefs = NULL, seed = NULL, ...)
```

**Arguments**

- **nStudents** integer indicating the number of students (in the college admissions problem) or men (in the stable marriage problem) in the market. Defaults to `ncol(s.prefs)`.
- **nColleges** integer indicating the number of colleges (in the college admissions problem) or women (in the stable marriage problem) in the market. Defaults to `ncol(c.prefs)`.
- **nSlots** vector of length `nColleges` indicating the number of places (i.e. quota) of each college. Defaults to `rep(1, nColleges)` for the marriage problem.
- **nCouples** integer indicating the number of couples (in the college admissions problem) or men (in the stable marriage problem) in the market. Defaults to `ncol(co.prefs)`.
- **s.prefs** matrix of dimension `nColleges x nStudents` with the `j`th column containing student `j`’s ranking over colleges in decreasing order of preference (i.e. most preferred first).
- **c.prefs** matrix of dimension `nStudents x nColleges` with the `i`th column containing college `i`’s ranking over students in decreasing order of preference (i.e. most preferred first).
- **co.prefs** matrix of dimension `4 x nCouplesPrefs` in long format with the `1`th and `2`th columns containing student couple id’s; `3`th and `4`th is a `2`-tuple ranking over college preference for the couple (coupleStudent1.pref, coupleStudent2.pref) in decreasing order of preference by rows (i.e. most preferred first).
- **seed** integer setting the state for random number generation.
- **...**
Value

hri2 returns a list of the following elements:

- `matchings`: list of matched students and colleges.
- `summary`: detailed report of the matching result, including further information on ranks.

See function: `summary.hrci(x)`

Minimum required arguments

hri2 requires the following combination of arguments, subject to the matching problem.

- `nStudents`, `nColleges`: Residence hospital problem without couples and random preferences.
- `nStudents`, `nColleges`, `nCouples`, `nSlots`: Residence hospital problem with couples and random preferences.
- `s.prefs`, `c.prefs`, `co.prefs`, `nSlots`: Residence hospital problem with couples and given preferences.

Author(s)

Sven Giegerich, Thilo Klein

References


Examples

```r
## Example with given preferences
s.prefs <- matrix(c(4,2,3,5,1, 2,1,3,4,5, 1,2,3,4,5), 5,3)
c.prefs <- matrix(c(1,2,3, 1,2,3, 1,2,3, 1,2,3, 1,2,3, 3,5))
co.prefs <- matrix(c(4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,1,2,1,2,3,3,4,3,1,1,2,1,2,3,3,3,4), 8,4)
res <- hri2(s.prefs=s.prefs, c.prefs=c.prefs, co.prefs=co.prefs, nSlots=c(1,1,1,1))
res$matchings
# summary(res)

## Example with random preferences
nStudents <- 50
nColleges <- 30
nCouples <- 4
nSlots <- sample(1:nStudents, nColleges)
res <- hri2(nStudents=nStudents, nColleges=nColleges, nCouples=nCouples, nSlots=nSlots)
res$matchings
# summary(res)
```
Immediate Acceptance Algorithm (a.k.a. Boston mechanism) for two-sided matching markets

Description

Finds the optimal assignment of students to colleges in the college admissions problem based on the Boston mechanism. The option acceptance="deferred" instead uses the Gale-Shapley (1962) Deferred Acceptance Algorithm with male offer. The function works with either given or randomly generated preferences.

Usage

iaa(nStudents = ncol(s.prefs), nColleges = ncol(c.prefs), nSlots = rep(1, nColleges), s.prefs = NULL, c.prefs = NULL, acceptance = "immediate")

Arguments

- nStudents: integer indicating the number of students (in the college admissions problem) or men (in the stable marriage problem) in the market. Defaults to ncol(s.prefs).
- nColleges: integer indicating the number of colleges (in the college admissions problem) or women (in the stable marriage problem) in the market. Defaults to ncol(c.prefs).
- nSlots: vector of length nColleges indicating the number of places (i.e. quota) of each college. Defaults to rep(1, nColleges) for the marriage problem.
- s.prefs: matrix of dimension nColleges x nStudents with the jth column containing student j’s ranking over colleges in decreasing order of preference (i.e. most preferred first).
- c.prefs: matrix of dimension nStudents x nColleges with the ith column containing college i’s ranking over students in decreasing order of preference (i.e. most preferred first).
- acceptance: if acceptance="deferred" returns the solution found by the student-proposing Gale-Shapley deferred acceptance algorithm; if acceptance="immediate" (the default) returns the solution found by the Boston mechanism.

Value

iaa returns a list with the following elements.

- s.prefs: student-side preference matrix.
- c.prefs: college-side preference matrix.
- iterations: number of interations required to find the stable matching.
- matchings: edgelist of matches
- singles: identifier of single (or unmatched) students/men.
Minimum required arguments

`iaa` requires the following combination of arguments, subject to the matching problem.

- `nStudents`, `nColleges` Marriage problem with random preferences.
- `s.prefs`, `c.prefs` Marriage problem with given preferences.
- `nStudents`, `nSlots` College admissions problem with random preferences.
- `s.prefs`, `c.prefs`, `nSlots` College admissions problem with given preferences.

Author(s)

Thilo Klein

References


Examples

```r
## ----------------------------------------
## College admission problem

## Boston mechanism for 7 students, 2 colleges with 3 slots each
set.seed(123)
iaa(nStudents=7, nSlots=c(3,3))

## Gale-Shapley algorithm
set.seed(123)
iaa(nStudents=7, nSlots=c(3,3), acceptance="deferred")

## Same results for the Gale-Shapley algorithm with hri() function
set.seed(123)
hri(nStudents=7, nSlots=c(3,3))

## 7 students, 2 colleges with 3 slots each, given preferences:
s.prefs <- matrix(c(1,2, 1,2, 1,2, 1,2, 1,2, 1,2, 1,2, 1,2, 2,7)
c.prefs <- matrix(c(1,2,3,4,5,6,7, 1,2,3,4,5,6,7), 7,2)
iaa(s.prefs=s.prefs, c.prefs=c.prefs, nSlots=c(3,3))
```

Karlson-Holm-Breen method for comparing probit coefficients

Description

Significance test for confounding; that is, the difference between regression coefficients from same-sample nested logit and probit models. The test procedure follows Karlson et al (2012), Section 3.4.
Usage

khb(X, y, z)

Arguments

X  data frame comprising independent variables including confounding variable.

y  vector of dependent variable.

z  character string giving the name of the confounding variable in X.

Author(s)

Thilo Klein

References


Examples

```r
## 1. load results from Klein (2015a)
data(klein15a)

## 2. apply KHB method
with(klein15a$variables, khb(x=X, y=Y, z="eta"))
```

klein15a  MCMC results in Klein (2015a)

Description

MCMC results in Klein (2015a).

Usage

data(klein15a)

Format

A list containing the following elements:

- **model.list**
- **coefs**

References

Results of Monte Carlo Simulations in Klein (2015b)

Description

Results of Monte Carlo Simulations in Klein (2015b) for 40 two-group markets.

Usage

data(klein15b)

Format

A list containing the following elements:

- **exp.5.5.ols**: Benchmark study, OLS: coefficient estimates for 40 markets with groups of 5. Data for all 5 group members is observed.
- **exp.5.5.ntu**: Benchmark study, structural model.
- **exp.6.5.ols**: Experiment 1, OLS: coefficient estimates for 40 markets with groups of 6. Only Data for 5 group members is observed.
- **exp.6.5.ntu**: Experiment 1, structural model.
- **exp.6.6.ols**: Experiment 2, OLS: coefficient estimates for 40 markets with groups of 6. Data for all 6 group members is observed but only a random sample of 250 of the 922 counterfactual groups is used in the analysis.
- **exp.6.6.ntu**: Experiment 2, structural model.

References


Examples

```r
## Plot of posterior distributions

data(klein15b)

tpe <- c(rep("Benchmark",2), rep("Experiment 1",2), rep("Experiment 2",2))

for(i in seq(1,length(klein15b)-1,2)){
  ntu <- klein15b[[i]]
  ols <- klein15b[[i+1]]

  ntu <- ntu[,colnames(ntu) == "beta.wst.ieq"]
  ols <- ols[,colnames(ols) == "beta.wst.ieq"]
```
if(i == 1){
  draws <- data.frame(Structural=ntu, OLS=ols, type=tpe[i]) #, stringsAsFactors=FALSE
} else{
  draws <- rbind(draws, data.frame(Structural=ntu, OLS=ols, type=tpe[i]))
}

library(lattice)
lattice.options(default.theme = standard.theme(color = FALSE))
keys <- list(text=c("Structural model","OLS"), space="top", columns=2, lines=TRUE)
densityplot(~ Structural + OLS | type, plot.points=FALSE, auto.key=keys,
  data = draws, xlab = "coefficient draws", ylab = "density", type = "l",
  panel = function(x,...) {
    panel.densityplot(x,...)
    panel.abline(v=-1, lty=3)
  })

## Not run:
## Modes of posterior distributions

## load data
data(klein15b)

## define function to obtain the mode
mode <- function(x){
  d <- density(x,bw="SJ")
  formatC(round(d$x[which.max(d$y)], 3), format='f', digits=3)
}

## Benchmark study
apply(klein15b$exp.5.5.ntu, 2, mode)
apply(klein15b$exp.5.5.ols, 2, mode)

## Experiment 1
apply(klein15b$exp.6.5.ntu, 2, mode)
apply(klein15b$exp.6.5.ols, 2, mode)

## Experiment 2
apply(klein15b$exp.6.6.ntu, 2, mode)
apply(klein15b$exp.6.6.ols, 2, mode)

## End(Not run)

mce                MC Experiments

Description
MC Experiments
Usage

mce(seed, niter, N, n, m, type, method)

Arguments

- **seed**: seed value for Monte Carlo Experiment
- **niter**: number of draws in estimation
- **N**: group size (population)
- **n**: group size (sample)
- **m**: number of markets
- **type**: type of the MC Experiment. Either `group.members` for randomly sampled group members or `counterfactual.groups` for randomly sampled number of counterfactual (or feasible) groups in selection equation (capped at limit max.combs=250)
- **method**: either `group.members` or `counterfactual.groups`

Author(s)

Thilo Klein

Examples

```r
## Not run:
## 1. Set parameters
mciter <- 2 #500
niter <- 10 #400000
nodes <- 4

## 2. Setup parallel backend to use 4 processors
library(foreach); library(doSNOW)
cl <- makeCluster(4); registerDoSNOW(cl)

## 3. Define foreach loop function
mce.add <- function(mciter, niter, N, n, m, type, method){
  h <- foreach(i=1:mciter) %dopar% {
    library(matchingMarkets)
    mce(seed=i, niter, N, n, m, type, method)
  }
do.call(rbind, h)
}

## 4. Run simulations:

## 4-a. Benchmark study
exp.5.5.ols <- mce.add(mciter=mciter, niter=niter, N=5, n=5, m=40, type="group.members", method="outcome")
exp.5.5.ntu <- mce.add(mciter=mciter, niter=niter, N=5, n=5, m=40, type="group.members", method="NTU")

## 4-b. Experiment 1: randomly sampled group members
```
plp

Partitioning Linear Programme for the stable roommates problem

Description


Usage

plp(V = NULL, N = NULL)

Arguments

V valuation matrix of dimension N\times N that gives row-players valuation over column players (or vice versa).

N integer (divisible by 2) that gives the number of players in the market.

Value

plp returns a list with the following items.

Valuation.matrix input values of V.

Assignment.matrix upper triangular matrix of dimension N\times N with entries of 1 for equilibrium pairs and 0 otherwise.

Equilibrium.groups matrix that gives the N/2 equilibrium pairs and equilibrium partners’ mutual valuations.
**Author(s)**

Thilo Klein

**References**


**Examples**

```r
## Roommate problem with 10 players, transferable utility and random preferences:
plp(N=10)

## Roommate problem with 10 players, transferable utility and given preferences:
V <- matrix(rep(1:10, 10), 10, 10)
plp(V=V)
```

**predict.stabit2**  
*Predict method for fitted matching models*

**Description**

Calculate predicted values for matching models fitted with functions `stabit` and `stabit2`.

**Usage**

```r
## S3 method for class 'stabit2'
predict(object, newdata = NULL, ...)
```

**Arguments**

- `object`: a fitted object of class `stabit`
- `newdata`: optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors or the fitted response values are returned.
- `...`: 

**Author(s)**

Thilo Klein

**References**

Examples

```r
## load the results from Klein (2015) paper
data(klein15a)

## predict the latent outcome variable
predict(klein15a)
```

---

**sri**

*All stable matchings in the stable roommates problem with incomplete lists*

---

**Description**

Finds all stable matchings (if one exists) in the stable roommates problem with incomplete lists using the Prosser (2014) constraint encoding based on either given or randomly generated preferences.

**Usage**

```r
sri(prefs = NULL, nAgents = NULL, seed = NULL, p.range = NULL)
```

**Arguments**

- `prefs` valuation matrix of dimension `nAgents x nAgents` that gives column-players’ ranking over other players in decreasing order of preference (i.e. most preferred first).
- `nAgents` integer that gives the number of players in the market.
- `seed` integer setting the state for random number generation.
- `p.range` range of two integers `p.range = c(p.min, p.max)`, where `p.min < p.max`. Produces incomplete preference lists with the length of each player’s list randomly sampled from the range `[p.min, p.max]`. Note: interval is always too small because non-permissible matches are automatically deleted.

**Value**

`sri` returns a list with the following items.

- `prefs` agents’ preference list.
- `matching` edgelist of matched pairs, including the number of the match (`matching`).

**Author(s)**

Thilo Klein
References


Examples

```r
## Roommate problem with 10 players, given preferences:
prefs <- matrix(rep(c(1:10, 10), 10), 10, 10)
sri(prefs=prefs)

## Roommate problem with 10 players, random preferences:
sri(nAgents=10, seed=1)

## Roommate problem with no equilibrium matching:
sri(nAgents=10, seed=2)

## Roommate problem with 3 equilibria:
sri(nAgents=10, seed=3)
```

---

stabit  Matching model and selection correction for group formation

Description

The function provides a Gibbs sampler for a structural matching model that estimates preferences and corrects for sample selection bias when the selection process is a one-sided matching game; that is, group/coalition formation.

The input is individual-level data of all group members from one-sided matching markets; that is, from group/coalition formation games.

In a first step, the function generates a model matrix with characteristics of all feasible groups of the same size as the observed groups in the market.

For example, in the stable roommates problem with \( n = 4 \) students \( \{1, 2, 3, 4\} \) sorting into groups of 2, we have \( \binom{4}{2} = 6 \) feasible groups: \((1,2)(3,4)\) \((1,3)(2,4)\) \((1,4)(2,3)\).

In the group formation problem with \( n = 6 \) students \( \{1, 2, 3, 4, 5, 6\} \) sorting into groups of 3, we have \( \binom{6}{3} = 20 \) feasible groups. For the same students sorting into groups of sizes 2 and 4, we have \( \binom{6}{2} + \binom{6}{4} = 30 \) feasible groups.

The structural model consists of a selection and an outcome equation. The Selection Equation determines which matches are observed \( (D = 1) \) and which are not \( (D = 0) \).

\[
D = 1 | V \in \Gamma \\
V = W \alpha + \eta
\]
Here, $V$ is a vector of latent valuations of all feasible matches, i.e., observed and unobserved, and $1[.]$ is the Iverson bracket. A match is observed if its match valuation is in the set of valuations $\Gamma$ that satisfy the equilibrium condition (see Klein, 2015a). This condition differs for matching games with transferable and non-transferable utility and can be specified using the `method` argument. The match valuation $V$ is a linear function of $W$, a matrix of characteristics for all feasible groups, and $\eta$, a vector of random errors. $\alpha$ is a parameter vector to be estimated.

The Outcome Equation determines the outcome for observed matches. The dependent variable can either be continuous or binary, dependent on the value of the `binary` argument. In the binary case, the dependent variable $R$ is determined by a threshold rule for the latent variable $Y$.

\[
R = 1[Y > c]
\]

\[
Y = X\beta + \epsilon
\]

Here, $Y$ is a linear function of $X$, a matrix of characteristics for observed matches, and $\epsilon$, a vector of random errors. $\beta$ is a parameter vector to be estimated.

The structural model imposes a linear relationship between the error terms of both equations as $\epsilon = \delta\eta + \xi$, where $\xi$ is a vector of random errors and $\delta$ is the covariance parameter to be estimated. If $\delta$ were zero, the marginal distributions of $\epsilon$ and $\eta$ would be independent and the selection problem would vanish. That is, the observed outcomes would be a random sample from the population of interest.

**Usage**

```r
stabit(x, m.id = "m.id", g.id = "g.id", R = "R", selection = NULL, outcome = NULL, simulation = "none", seed = 123, max.combs = Inf, method = "NTU", binary = FALSE, offsetOut = 0, offsetSel = 0, marketFE = FALSE, censored = 0, gPrior = FALSE, dropOnes = FALSE, interOut = 0, interSel = 0, standardize = 0, niter = 10, verbose = FALSE)
```

**Arguments**

- `x`: data frame with individual-level characteristics of all group members including market- and group-identifiers.
- `m.id`: character string giving the name of the market identifier variable. Defaults to "m.id".
- `g.id`: character string giving the name of the group identifier variable. Defaults to "g.id".
- `R`: dependent variable in outcome equation. Defaults to "R".
- `selection`: list containing variables and pertaining operators in the selection equation. The format is operation = "variable". See the Details and Examples sections.
- `outcome`: list containing variables and pertaining operators in the outcome equation. The format is operation = "variable". See the Details and Examples sections.
- `simulation`: should the values of dependent variables in selection and outcome equations be simulated? Options are "none" for no simulation, "NTU" for non-transferable utility matching, "TU" for transferable utility or "random" for random matching of individuals to groups. Simulation settings are (i) all model coefficients set to
alpha=beta=1; (ii) covariance between error terms delta=0.5; (iii) error terms eta and xi are draws from a standard normal distribution.

seed
integer setting the state for random number generation if simulation=TRUE.

max.combs
integer (divisible by two) giving the maximum number of feasible groups to be used for generating group-level characteristics.

method
estimation method to be used. Either "NTU" or "TU" for selection correction using non-transferable or transferable utility matching as selection rule; "outcome" for estimation of the outcome equation only; or "model.frame" for no estimation.

binary
logical: if TRUE outcome variable is taken to be binary; if FALSE outcome variable is taken to be continuous.

offsetOut
vector of integers indicating the indices of columns in X for which coefficients should be forced to 1. Use 0 for none.

offsetSel
vector of integers indicating the indices of columns in W for which coefficients should be forced to 1. Use 0 for none.

marketFE
logical: if TRUE market-level fixed effects are used in outcome equation; if FALSE no market fixed effects are used.

censored
draws of the delta parameter that estimates the covariation between the error terms in selection and outcome equation are 0: not censored, 1: censored from below, 2: censored from above.

gPrior
logical: if TRUE the g-prior (Zellner, 1986) is used for the variance-covariance matrix.

dropOnes
logical: if TRUE one-group-markets are excluded from estimation.

interOut
two-column matrix indicating the indices of columns in X that should be interacted in estimation. Use 0 for none.

interSel
two-column matrix indicating the indices of columns in W that should be interacted in estimation. Use 0 for none.

standardize
numeric: if standardize>0 the independent variables will be standardized by dividing by standardize times their standard deviation. Defaults to no standardization standardize=0.

niter
number of iterations to use for the Gibbs sampler.

verbose
.

Details
Operators for variable transformations in selection and outcome arguments.

add
sum over all group members and divide by group size.

int
sum over all possible two-way interactions x * y of group members and divide by the number of those, given by choose(n, 2).

ieq
sum over all possible two-way equality assertions 1[x = y] and divide by the number of those.

ive
sum over all possible two-way interactions of vectors of variables of group members and divide by number of those.

inv ...

dst
sum over all possible two-way distances between players and divide by number of those, where distance is defined as e^{-[x-y]}. 


Author(s)
Thilo Klein

References


Examples

```r
## Not run:
## --- SIMULATED EXAMPLE ---

## 1. Simulate one-sided matching data for 200 markets (m=200) with 2 groups
## per market (gpm=2) and 5 individuals per group (ind=5). True parameters
## in selection equation is wst=1, in outcome equation wst=0.

## 1-a. Simulate individual-level, independent variables
idata <- stabsim(m=200, ind=5, seed=123, gpm=2)
head(idata)

## 1-b. Simulate group-level variables
mdata <- stabit(x=idata, simulation="NTU", method="model.frame",
selection = list(add="wst"), outcome = list(add="wst"), verbose=FALSE)
head(mdata$OUT)
head(mdata$SEL)

## 2. Bias from sorting

## 2-a. Naive OLS estimation
lm(R ~ wst.add, data=mdata$OUT)$coefficients

## 2-b. epsilon is correlated with independent variables
with(mdata$OUT, cor(epsilon, wst.add))

## 2-c. but xi is uncorrelated with independent variables
with(mdata$OUT, cor(xi, wst.add))

## 3. Correction of sorting bias when valuations V are observed

## 3-a. 1st stage: obtain fitted value for eta
lm.sel <- lm(V ~ -1 + wst.add, data=mdata$SEL)
lm.sel$coefficients
eta <- lm.sel$resid[mdata$SEL$D==1]

## 3-b. 2nd stage: control for eta
lm(R ~ wst.add + eta, data=mdata$OUT)$coefficients
```
Matching model and selection correction for college admissions

Description

The function provides a Gibbs sampler for a structural matching model that estimates preferences and corrects for sample selection bias when the selection process is a two-sided matching game; i.e., a matching of students to colleges.

The structural model consists of a selection and an outcome equation. The Selection Equation determines which matches are observed ($D = 1$) and which are not ($D = 0$).

$$ D = 1[V \in \Gamma] $$

$$ V = W\beta + \eta $$

Here, $V$ is a vector of latent valuations of all feasible matches, i.e., observed and unobserved, and $1[.]$ is the Iverson bracket. A match is observed if its match valuation is in the set of valuations $\Gamma$ that
satisfy the equilibrium condition (see Sorensen, 2007). The match valuation $V$ is a linear function of $W$, a matrix of characteristics for all feasible matches, and $\eta$, a vector of random errors. $\beta$ is a parameter vector to be estimated.

The Outcome Equation determines the outcome for observed matches. The dependent variable can either be continuous or binary, dependent on the value of the binary argument. In the binary case, the dependent variable $R$ is determined by a threshold rule for the latent variable $Y$.

$$R = 1[Y > c]$$

$$Y = X\alpha + \epsilon$$

Here, $Y$ is a linear function of $X$, a matrix of characteristics for observed matches, and $\epsilon$, a vector of random errors. $\alpha$ is a parameter vector to be estimated.

The structural model imposes a linear relationship between the error terms of both equations as $\epsilon = \kappa\eta + \nu$, where $\nu$ is a vector of random errors and $\kappa$ is the covariance parameter to be estimated. If $\kappa$ were zero, the marginal distributions of $\epsilon$ and $\eta$ would be independent and the selection problem would vanish. That is, the observed outcomes would be a random sample from the population of interest.

**Usage**

```r
stabit2(OUT = NULL, SEL = NULL, colleges = NULL, students = NULL,
        outcome = NULL, selection, binary = FALSE, niter, gprior = FALSE,
        censored = 1, thin = 1, nCores = max(1, detectCores() - 1), ...)`
```

**Arguments**

- **OUT** data frame with characteristics of all observed matches, including market identifier m.id, college identifier c.id and student identifier s.id.
- **SEL** optional: data frame with characteristics of all observed and unobserved matches, including market identifier m.id, college identifier c.id and student identifier s.id.
- **colleges** character vector of variable names for college characteristics. These variables carry the same value for any college.
- **students** character vector of variable names for student characteristics. These variables carry the same value for any student.
- **outcome** formula for match outcomes.
- **selection** formula for match valuations.
- **binary** logical: if TRUE outcome variable is taken to be binary; if FALSE outcome variable is taken to be continuous.
- **niter** number of iterations to use for the Gibbs sampler.
- **gprior** logical: if TRUE the g-prior (Zellner, 1986) is used for the variance-covariance matrix. (Not yet implemented)
- **censored** draws of the kappa parameter that estimates the covariance between the error terms in selection and outcome equation are 0:not censored, 1:censored from below, 2:censored from above.
stabit2

integer indicating the level of thinning in the MCMC draws. The default thin=1 saves every draw, thin=2 every second, etc.

nCores number of cores to be used in parallel Gibbs sampling.

... 

Author(s)

Thilo Klein

References


Examples

```r
## Not run:
## --- SIMULATED EXAMPLE ---

## 1. Simulate two-sided matching data for 20 markets (m=20) with 100 students
## (nStudents=100) per market and 20 colleges with quotas of 5 students, each
## (nSlots=rep(5,20)). True parameters in selection and outcome equations are
## all equal to 1.

xdata <- stabsim2(m=20, nStudents=100, nSlots=rep(5,20), verbose=FALSE, 
  colleges = "c1", students = "s1", 
  outcome = ~ c1:s1 + eta + nu, 
  selection = ~ -1 + c1:s1 + eta
)
head(xdata$OUT)

## 2. Correction for sorting bias when match valuations V are observed

## 2-a. Bias from sorting
lm1 <- lm(y ~ c1:s1, data=xdata$OUT)
summary(lm1)

## 2-b. Cause of the bias
with(xdata$OUT, cor(c1*s1, eta))

## 2-c. Correction for sorting bias
lm2a <- lm(V ~ -1 + c1:s1, data=xdata$SEL); summary(lm2a)
etahat <- lm2a$residuals[xdata$SEL$D==1]

lm2b <- lm(y ~ c1:s1 + etahat, data=xdata$OUT)
summary(lm2b)

## 3. Correction for sorting bias when match valuations V are unobserved

## 3-a. Run Gibbs sampler (when SEL is given)
```r
fit2 <- stabit2(OUT = xdata$OUT, 
    SEL = xdata$SEL, 
    outcome = y ~ c1:s1, 
    selection = ~ -1 + c1:s1, 
    niter=1000
)
```

```r
## 3-b. Alternatively: Run Gibbs sampler (when SEL is not given)
fit2 <- stabit2(OUT = xdata$OUT, 
    colleges = "c1", 
    students = "s1", 
    outcome = y ~ c1:s1, 
    selection = ~ -1 + c1:s1, 
    niter=1000
)
```

```r
## 4. Implemented methods
## 4-a. Get coefficients
fit2

## 4-b. Coefficient table
summary(fit2)

## 4-c. Get marginal effects
summary(fit2, mfx=TRUE)

## 4-d. Also try the following functions
#coef(fit2) 
#fitted(fit2) 
#residuals(fit2) 
#predict(fit2, newdata=NULL)
```

```r
## 5. Plot MCMC draws for coefficients
plot(fit2)
```

## End(Not run)

---

**stabsim**

*Simulated data for group formation problem*

---

**Description**

Simulate individual-level data for one-sided matching markets.

**Usage**

```r
stabsim(m, ind, seed = 123, singles = NULL, gpm = 2)
```
**Arguments**

- `m` integer indicating the number of markets to be simulated.
- `ind` integer (or vector) indicating the number of individuals per group.
- `seed` integer setting the state for random number generation. Defaults to `set.seed(123)`.
- `singles` integer giving the number of one-group markets.
- `gpm` integer giving the number of groups per market.

**Value**

`stabsim` returns a data frame with the randomly generated variables mimicking those in dataset `baac00`.

- `m.id` categorical: market identifier.
- `g.id` categorical: group identifier.
- `wst` binary: indicator taking the value 1 if last year was worse than the year before; 0 otherwise.
- `R` NA: group outcome is not simulated. It can be obtained using the simulation argument in function `stabit`.

**Author(s)**

Thilo Klein

**Examples**

```r
## Coalitions [gpm := 2 !]
## Simulate one-sided matching data for 4 markets (m=4) with 2 groups
## per market (gpm=2) and 2 to 4 individuals per group (ind=2:4)
idata <- stabsim(m=4, ind=2:4, seed=124, singles=2, gpm=2)

## Roommates [ind := 2 !]
## Simulate one-sided matching data for 3 markets (m=3) with 3 groups
## per market (gpm=3) and 2 individuals per group (ind=2)
idata <- stabsim(m=3, ind=2, seed=124, gpm=3)
```

---

**Description**

Simulate data for two-sided matching markets. In the simulation for the Sorensen (2007) model with one selection equation, an equal sharing rule of $\lambda = 0.5$ is used.

**Usage**

```r
stabsim2(m, nStudents, nColleges = length(nSlots), nSlots, colleges, students,
          outcome, selection, binary = FALSE, seed = 123, verbose = TRUE)
```
Arguments

- `m`: integer indicating the number of markets to be simulated.
- `nStudents`: integer indicating the number of students per market.
- `nColleges`: integer indicating the number of colleges per market.
- `nSlots`: vector of length `nColleges` indicating the number of places at each college, i.e. the college’s quota.
- `colleges`: character vector of variable names for college characteristics. These variables carry the same value for any college.
- `students`: character vector of variable names for student characteristics. These variables carry the same value for any student.
- `outcome`: formula for match outcomes.
- `selection`: formula for match valuations.
- `binary`: logical: if TRUE outcome variable is binary; if FALSE outcome variable is continuous.
- `seed`: integer setting the state for random number generation. Defaults to `set.seed(123)`.
- `verbose`: integer setting the state for random number generation. Defaults to `set.seed(123)`.

Value

`stabsim2` returns a list with the following items.

- `OUT`
- `SEL`
- `SELC`
- `SELS`

Author(s)

Thilo Klein

Examples

```r
## Simulate two-sided matching data for 2 markets (m=2) with 10 students
## (nStudents=10) per market and 3 colleges (nColleges=3) with quotas of
## 2, 3, and 5 students, respectively.

xdata <- stabsim2(m=2, nStudents=10, nSlots=c(2,3,5), verbose=FALSE,
colleges = "c1", students = "s1",
outcome = - c1:s1 + eta + nu,
selection = -1 + c1:s1 + eta
)
head(xdata$OUT)
head(xdata$SEL)
```
Top-Trading-Cycles Algorithm for the house allocation problem

Description


Usage

\texttt{ttc(P = NULL, X = NULL)}

Arguments

- \texttt{P} \hspace{1cm} \text{matrix of individuals' preference rankings (Rank Order Lists) over objects.}
- \texttt{X} \hspace{1cm} \text{2-column-matrix of objects (\texttt{obj}) and their owners (\texttt{ind}).}

Value

\texttt{ttc} returns a 2-column matrix of the matching solution for the housing market problem based on the Top-Trading-Cycles algorithm.

Author(s)

Thilo Klein

References


Examples

```r
## generate matrix of individuals' preference rankings over objects, a.k.a. Rank Order Lists (ROL)
P <\texttt{\textless \textgreater} \texttt{matrix(c(2, 5, 1, 4, 3, \# ROL of individual 1}}
1, 5, 4, 3, 2, \# ind 2
2, 1, 4, 3, 5, \# ind 3
2, 4, 3,NA,NA, \# ind 4
4, 3, 1, 2,NA), \# ind 5
byrow=FALSE, nrow=5); P

## generate 2-column-matrix of objects ('\texttt{obj}') and their owners ('\texttt{ind}')
X <\texttt{\textless \textgreater} \texttt{data.frame(ind=1:5, obj=5:1); X}

## find assignment based on TTC algorithm
\texttt{ttc(P=P, X=X)}
```
Index

*Topic **algorithms**, hri2, 8
  *Topic **hri**
    hri, 5
    iaa, 10
    plp, 16
    sri, 18
    ttc, 29
*Topic **matching**
  hri2, 8
*Topic **package**
  matchingMarkets-package, 2
*Topic **regression**
  stabit, 19
  stabitz2, 23
*Topic **summary**
  khb, 11
  predict.stabit2, 17

baac00, 3, 4, 27
hri, 3, 5
hri2, 8
iaa, 10

khh, 2, 11
klein15a, 12
klein15b, 13

matchingMarkets
  (matchingMarkets-package), 2
matchingMarkets-package, 2
mce, 14

plp, 16
predict.stabit2, 17
sri, 3, 18
stabit, 2, 19
stabitz2, 2, 23
stabsim, 3, 26
stabsim2, 3, 27
ttc, 3, 29