

Package ‘ipr’

September 11, 2026

Type Package

Title Iterative Proportional Repartition Algorithm

Version 1.0.0

Maintainer Jean-Benoit Rossel <jean-benoit.rossel@unisante.ch>

Description Let us consider a sample of patients who can suffer from several diseases simultaneously, in a given set of diseases. The goal of the implemented algorithm is to estimate the individual average cost of each disease, starting from the global health costs available for each patient.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Jean-Benoit Rossel [aut, cre],
Valentin Rousson [aut],
Yves Egli [aut]

Repository CRAN

Date/Publication 2026-09-11 10:10:02 UTC

Contents

ipr	1
ipr_boot	4

Index	7
--------------	----------

ipr	<i>Iterative Proportional Repartition (IPR) algorithm</i>
-----	---

Description

Estimating the health cost repartition among diseases in the presence of multimorbidity, i.e. when some patients have multiple diseases. Using the Iterative Proportional Repartition algorithm (see reference below), the goal is to estimate the average cost for each disease, starting from the global health costs available for each patient.

Usage

```
ipr(X, y, print.it=FALSE, start=rep(1,dim(X)[2]), cutup=Inf, cutlow=cutup,
epsrel=0.001, epsabs=0.1, maxiter=1000, det=FALSE)
```

Arguments

<code>X</code>	Matrix with $x_{ij} = 1$ if patient i suffers from disease j and $x_{ij} = 0$ otherwise. Each row thus refers to one patient and each column to one disease. The number of columns of <code>X</code> corresponds to the number of diseases considered.
<code>y</code>	Vector where y_i is the global health cost of patient i . The length of <code>y</code> must be equal to the number of rows of <code>X</code> .
<code>print.it</code>	Logical. If TRUE, the number of the current iteration and the current estimates are printed.
<code>start</code>	Vector of initial estimates of the average cost for each disease to start IPR algorithm. Default is an initial average cost of 1 for all diseases. The length of <code>start</code> must be equal to the number of columns of <code>X</code> .
<code>cutup, cutlow</code>	Options which can be used to get a robust version of IPR. If the current allocated cost of disease j for patient i is more than <code>cutup</code> times more expansive (or less than <code>cutlow</code> times less expansive) than the current average cost estimate of that disease j , then this outlying allocated cost is not taken into account in the next iteration to compute the average cost of disease j . By default, <code>cutup</code> and <code>cutlow</code> are set to Inf.
<code>epsrel</code>	Stopping criterion such that the IPR algorithm stops if for all diseases, the current estimated average cost differs by less than $100 \cdot \text{epsrel}$ percent from what it was at the previous iteration. The default value is 0.001. Should be set to 0 to ignore that criterion.
<code>epsabs</code>	Stopping criterion such that the IPR algorithm stops if for all diseases, the current estimated average cost differs (in absolute value) by less than <code>epsabs</code> percent from what it was at the previous iteration. The default value is 0.1. Should be set to 0 to ignore that criterion.
<code>maxiter</code>	Maximal number of iterations of IPR algorithm. The default value is 1000.
<code>det</code>	Logical. If TRUE, the allocated costs of each disease for each patient are given, by returning a matrix Y where y_{ij} is the estimated cost of disease j for patient i .

Details

Let us consider n patients and p diseases. We are given a matrix X such that $x_{ij} = 1$ if the patient i suffers from disease j and $x_{ij} = 0$ otherwise. We are also given a vector y , where y_i is the global health cost of patient i . In order to estimate the average cost of each disease, the IPR algorithm works as follows:

1. Start with some initial estimates μ_j , e.g. $\mu_j = 1$ for all $j = 1, \dots, p$. Those initial estimates are stored in the vector `start`.
2. Allocate the cost y_i among the diseases diagnosed for patient i , proportionally to the current estimates μ_j .

3. Update the current estimate of μ_j by averaging the specific costs obtained in step 2 for the disease j over the patients having that disease.
4. Repeat steps 2 and 3 until a stopping criterion, based on relative or absolute distance between two consecutive iterations. The stopping criterion can be defined with `epsabs` or `epsrel`.

By construction, the IPR algorithm satisfies two properties. First, it allows to obtain positive estimates for each average disease cost. Secondly, it allows to retrieve the total health costs. In other words, the sum of the estimates μ_j multiplied by the number of patients suffering from j is equal to the sum of the costs y_i .

The estimate of total cost τ_j spent for disease j as well as the estimated proportion π_j of the total costs which is allocated to disease j are also returned by our function. Mathematically, $\tau_j = \sum_{i=1}^n X_{ij} \cdot \mu_j$, while π_j is defined by $\pi_j = \tau_j / \sum_{k=1}^p \tau_k$.

Confidence intervals for each μ_j may be obtained with the `ipr_boot` function, which is based on bootstrapping technique.

Value

<code>coef</code>	A vector with the estimated average cost μ_j of each disease.
<code>total</code>	A vector with the estimated total cost τ_j spent for each disease.
<code>proportions</code>	A vector with the estimated proportion π_j of total cost spent for each disease.
<code>niter</code>	The number of iterations of IPR algorithm until the stopping criterion is achieved.
<code>epsrel</code>	The stopping criterion based on a relative distance between two consecutive iterations which has been used.
<code>epsabs</code>	The stopping criterion based on an absolute distance between two consecutive iterations which has been used.
<code>detail</code>	A matrix with the allocated costs of each disease for each patient, if <code>det</code> is set to TRUE.

Author(s)

Jean-Benoit Rossel (<jean-benoit.rossel@unisante.ch>), Valentin Rousson, and Yves Egli.

References

- Rousson, V., Rossel, J.-B. & Egli, Y. (2019). Estimating Health Cost Repartition Among Diseases in the Presence of Multimorbidity. *Health Services Research and Managerial Epidemiology*, 6.
- Rossel, J.-B., Rousson, V. & Egli, Y. (2021). A comparison of statistical methods for allocating disease costs in the presence of interactions. *Statistics in Medicine*, 40(14).

See Also

[ipr_boot](#).

Examples

```
# Here is a first example with 10 patients and 4 diseases:
X <- matrix(c(1,0,0,0,
0,1,1,0,
0,1,0,1,
1,0,0,1,
1,1,1,0,
0,0,1,1,
0,1,0,0,
1,1,0,0,
0,1,1,1,
0,0,0,1),ncol=4,byrow=TRUE)

y <- c(500,200,100,400,1000,500,100,300,800,2000)

# If we would use a linear model without intercept to estimate the average
# disease costs, we would obtain a negative value for disease 2.
lm(y~X-1)

# The IPR algorithm provides only positive estimates
ipr(X,y)

# Here is a second example:
X <- matrix(c(1,0,0,1,1,1),nrow=3,byrow=TRUE)
y <- c(5000,500,6600)

# We have three patients. The first one has only disease 1 with a cost of 5000.
# The second one has only disease 2 with a cost of 500 (i.e. ten times less
# expansive than disease 1). The third patient has both diseases with
# a cost of 6600 (i.e. 5000 + 500 + an extra cost of 1100).

# Using a linear model, one would allocate the extra cost equally between
# the three patients. The estimated average cost would thus be 5000+(1100/3)
# for disease 1 and 500+(1100/3) for disease 2.
lm(y~X-1)

# Using IPR algorithm, one allocates the extra cost taking into account that
# disease 1 is ten times more expansive than disease 2 when occurring alone.
# One thus gets an estimated average cost of 5500 for disease 1 and
# of 550 for disease 2.
ipr(X,y)
```

ipr_boot

*Bootstrap Confidence Intervals for Iterative Proportional Repartition
(IPR) algorithm*

Description

Obtaining confidence intervals for IPR estimates via bootstrapping. Three types of bootstrap methods are implemented.

Usage

```
ipr_boot(X, y, method = "normal", conf.level = 0.95, B = 100, ...)
```

Arguments

X	Matrix with $x_{ij} = 1$ if patient i suffers from disease j and $x_{ij} = 0$ otherwise. Each row thus refers to one patient and each column to one disease. The number of columns of X corresponds to the number of diseases considered.
y	Vector where y_i is the global health cost of patient i . The length of y must be equal to the number of rows of X.
method	A string representing the type of intervals to be computed. The value can be normal (default), basic, or perc, in order to get the normal, basic, or percentile bootstrap intervals, respectively. See details below. A value of all can also be used, to get the three intervals.
conf.level	A scalar containing the confidence level. Default value is 0.95.
B	Number of resamplings to get the intervals. Default value is 100.
...	Other arguments of the ipr function.

Details

Let $\hat{\mu}$ be a p -dimensional vector with the estimated costs of the diseases, obtained via the IPR algorithm. The function `ipr_boot` computes confidence intervals for these costs, via a bootstrapping technique. We draw B bootstrap samples and apply the IPR algorithm on each of them. Three possibilities are then implemented to get confidence intervals.

The *normal* bootstrap is based on a normal distribution of the estimator. We compute $\hat{\sigma}$, a vector with the estimated standard errors of the costs of each disease, using our B samples. Confidence intervals are then obtained by computing $\hat{\mu} \pm z_{1-\alpha/2} \cdot \hat{\sigma}$.

The two other methods are computed with the $\alpha/2$ - and $1 - \alpha/2$ -quantiles of the bootstrap distribution. Let us notice them by $q_{\alpha/2}$ and $q_{1-\alpha/2}$. The *percentile* bootstrap is given by these two quantiles (see Davison and Hinkley 1997, Equation (5.6) on page 194). The *basic* bootstrap is given by the equations $2\hat{\mu} - q_{1-\alpha/2}$ for the lower bound, and $2\hat{\mu} - q_{\alpha/2}$ for the upper bound (see Davison and Hinkley 1997, Equation (5.18) on page 203).

Considering $B = 100$ bootstrap samples may be enough with the normal bootstrap. However, higher values should be used with percentile or basic bootstrap.

Since the bootstrap samples are randomly defined, the user should define a random seed (e.g., `set.seed(1234)`) before using `ipr_boot`.

Value

estimate	A vector with the estimated average cost of each disease.
lower	A vector with the lower bounds of the confidence intervals.
upper	A vector with the upper bounds of the confidence intervals.
lower_normal	If the method is all, lower bounds obtained via normal bootstrap.
upper_normal	If the method is all, upper bounds obtained via normal bootstrap.

lower_perc	If the method is all, lower bounds obtained via percentile bootstrap.
upper_perc	If the method is all, upper bounds obtained via percentile bootstrap.
lower_basic	If the method is all, lower bounds obtained via basic bootstrap.
upper_basic	If the method is all, upper bounds obtained via basic bootstrap.

Author(s)

Jean-Benoit Rossel (<jean-benoit.rossel@unisant.ch>), Valentin Rousson, and Yves Egli.

References

- Davison, A. C., & Hinkley, D. V. (1997). Bootstrap methods and their application. *Cambridge university press*.
- Rousson, V., Rossel, J.-B. & Egli, Y. (2019). Estimating Health Cost Repartition Among Diseases in the Presence of Multimorbidity. *Health Services Research and Managerial Epidemiology*, 6.
- Rossel, J.-B., Rousson, V. & Egli, Y. (2021). A comparison of statistical methods for allocating disease costs in the presence of interactions. *Statistics in Medicine*, 40(14).

See Also

[ipr](#)

Examples

```
# Simulation of data with 30 patients and 7 diseases.
set.seed(123456)
n <- 30
p <- 7
X <- matrix(0,n,p,byrow=TRUE)
for(i in 1:n) X[i,] <- rbinom(p, size=1, prob=0.5)

# Simulation of costs. We consider lognormally distributed prices.
mean <- 1000
sdlog <- 0.2
cost <- matrix(rlnorm(n*p,log(mean)-sdlog^2/2,sdlog),n,p,byrow=TRUE)
y <- apply(X*cost,1,sum)

# Here are the estimated costs provided by IPR algorithm,
# and confidence intervals based on bootstrap techniques:
ipr_boot(X,y,method="all",B=1000)
```

Index

* **methods**

ipr, [1](#)

ipr_boot, [4](#)

* **models**

ipr, [1](#)

ipr_boot, [4](#)

ipr, [1](#), [6](#)

ipr_boot, [3](#), [4](#)