

# Package: PropScrRand (via r-universe)

March 3, 2025

**Type** Package

**Title** Propensity Score Methods for Assigning Treatment in Randomized Trials

**Version** 1.1.2

**Date** 2024-04-19

**Author** Travis Loux

**Maintainer** Travis Loux <travis.loux@slu.edu>

**Description** Contains functions to run propensity-biased allocation to balance covariate distributions in sequential trials and propensity-constrained randomization to balance covariate distributions in trials with known baseline covariates at time of randomization. Currently only supports trials comparing two groups.

**License** GPL-3

**Repository** <https://tloux.r-universe.dev>

**RemoteUrl** <https://github.com/tloux/propscrand>

**RemoteRef** HEAD

**RemoteSha** a50ec114d6e2c912cc3ba149054d126c94d0b80e

## Contents

genPerms . . . . .	2
getVar . . . . .	3
pba . . . . .	3
pcr . . . . .	5
piFunction . . . . .	6
plotpi . . . . .	7

<b>Index</b>	<b>9</b>
--------------	----------

---

genPerms                      *Generate Treatment Permutations*

---

### Description

Used within calls to pcr to generate a set of unique treatment permutations for randomization.

### Usage

```
genPerms(n, n1, nPerms)
```

### Arguments

n	Total number of units to be randomized.
n1	Number of units to receive treatment.
nPerms	Number of permutations to generate.

### Details

This function randomly samples nPerms of the choose(n, n1) possible treatment permutations. If nPerms > choose(n, n1), then all choose(n, n1) permutations are generated systematically. Also, in the case of 1-to-1 allocation, the complement treatment vectors are also produced, so the returned matrix has 2\*nPerms permutations. Uniqueness is checked throughout and duplicate permutations discarded.

### Value

The result is an n1 x nPerms (or n1 x choose(n, n1) or n1 x 2\*nPerms) matrix. Each column represents one treatment permutation, with the values in the column giving the index of the treated units.

### Author(s)

Travis M. Loux

### Examples

```
genPerms(n=50, n1=25, nPerms=500)  
genPerms(n=50, n1=35, nPerms=500)
```

---

getVar                                      *Compute Propensity Score Variance*

---

**Description**

Given a data set and vector of indices for treated units, computes the variance of the propensity score fitted via logistic regression.

**Usage**

```
getVar(covs, tIndex)
```

**Arguments**

covs                                      A data frame of baseline covariates.  
tIndex                                    A vector indicating which units are to receive treatment.

**Value**

Returns the variance of the fitted propensity scores.

**Author(s)**

Travis M. Loux

---

pba    *Propensity-Biased Allocation*

---

**Description**

Performs propensity-biased allocation for assigning a new unit to treatment in a sequential design with two treatment levels (i.e., treatment and control).

**Usage**

```
pba(x, tr, newx, k = 1, global = 0.5)  
pbaAgain(previous, newx, k = NA)
```

**Arguments**

x    A data frame of the covariate values of previously assigned units.  
tr    A vector of treatment assignments (0 or 1) for previously assigned units.  
newx                                        Data frame of covariate values of the new unit.  
k    Balancing parameter.  
global                                      Global target proportion to be treated.  
previous                                    The output of a previous call to pba or pbaAgain

## Details

The function `pba` generates a treatment assignment for a new unit. The steps of the process include regressing `tr` on `x` by logistic regression, computing the fitted value of the new unit using covariate values in `newx`, and transforming the fitted propensity score into the probability of treatment by a call to `piFunction` using `k` and `global` as parameters. The balancing parameter `k` must be one of  $0$ ,  $\text{Inf}$ , or the ratio of two positive odd integers. Small values of `k` result in less restrictive randomization while larger values of `k` result in more forced balance. In particular, `k = 0` is equivalent to pure randomization and `k = Inf` results in deterministic allocation. Finally, a treatment assignment for the new unit is generated via a Bernoulli trial with probability from `piFunction`.

The function `pbaAgain` takes as input the output from a previous call to `pba` or `pbaAgain` and runs `pba` for the new unit using the values of `newx`. If `k = NA` (the default), the value of `k` from previous is used; otherwise, the provided value of `k` is used. The parameter `global` is assumed to stay the same throughout the trial. The output of `pbaAgain` contains the same information as `pba`.

## Value

<code>results</code>	A list of results from the PBA procedure.
<code>phat</code>	The fitted propensity score for the new unit.
<code>ptreat</code>	The probability of assignment to the treatment group for the new unit.
<code>newtr</code>	Result of random assignment using <code>ptreat</code> .
<code>input</code>	A list of inputs to PBA procedure. Used in future calls to <code>pbaAgain</code> .
<code>x</code>	Input <code>x</code> .
<code>tr</code>	Input <code>tr</code> .
<code>newx</code>	Input <code>newx</code> .
<code>k</code>	Input <code>k</code> .
<code>global</code>	Input <code>global</code> .

## Author(s)

Travis Loux

## References

Loux, T.M. (2013) A simple, flexible, and effective covariate-adaptive treatment allocation procedure. *Statistics in Medicine* 32(22), 3775-3787. DOI: 10.1002/sim.5837

## Examples

```
x0 = data.frame(matrix(rnorm(60), ncol=3))
t0 = rbinom(nrow(x0), size=1, prob=0.5)

x1 = data.frame(matrix(rnorm(3), ncol=3))
trial1 = pba(x=x0, tr=t0, newx=x1, k=Inf)

x2 = data.frame(matrix(rnorm(3), ncol=3))
trial2 = pbaAgain(previous=trial1, newx=x2)
```

```
x3 = data.frame(matrix(rnorm(3), ncol=3))
trial3 = pbaAgain(previous=trial2, newx=x3, k=5/3)
```

pcr

*Propensity-Constrained Randomization***Description**

Performs propensity-constrained randomization on a given data set with measured covariates on all units.

**Usage**

```
pcr(x, nTreat, M, m)
```

**Arguments**

x	Data frame of covariates.
nTreat	Number of units to be treated.
M	Number of candidate permutations to create.
m	Number of permutations to keep.

**Details**

Given the parameters, `pcr` generates  $M$  unique permutations by calling `genPerms`. For each permutation, the empirical propensity scores are computed and the variance returned by `getVar`. Finally, the  $m$  permutations with the smallest propensity score variance are found. The  $m$  permutations returned in `best` can then be used to perform randomization and randomization inference. We suggest  $M \geq 10000$  and  $m/M \leq 0.10$ .

**Value**

treatments	The (approximately) $M$ permutations generated by <code>genPerms</code> .
variance	A vector of the propensity score variances for all $M$ permutations in treatments.
cutoff	The calculated $m/M$ quantile of propensity score variances.
best	The column indices of the permutations in treatments with propensity score variance below cutoff.

**Author(s)**

Travis Loux

**References**

Loux, T.M. (2015) Randomization, matching, and propensity scores in the design and analysis of experimental studies with known covariates. *Statistics in Medicine*. 34(4), 558-570. DOI: 10.1002/sim.6361

**Examples**

```

# 1:1 allocation, M small
dat1 = data.frame(x1=rnorm(50),
                  x2=rnorm(50),
                  x3=sample(c('a','b','c'), size=50, replace=TRUE))
trial1 = pcr(x=dat1, nTreat=25, M=500, m=50)

# 1:1 allocation, M large
dat2 = data.frame(x1=rnorm(10),
                  x2=rnorm(10),
                  x3=sample(c('a','b','c'), size=10, replace=TRUE))
trial2 = pcr(x=dat2, nTreat=5, M=200, m=20)

# non-1:1 allocation, M small
dat3 = data.frame(x1=rnorm(50),
                  x2=rnorm(50),
                  x3=sample(c('a','b','c'), size=50, replace=TRUE))
trial3 = pcr(x=dat3, nTreat=35, M=200, m=20)

# non-1:1 allocation, M large
dat4 = data.frame(x1=rnorm(10),
                  x2=rnorm(10),
                  x3=sample(c('a','b','c'), size=10, replace=TRUE))
trial4 = pcr(x=dat4, nTreat=6, M=300, m=30)

```

---

piFunction

*Get PBA Treatment Probability*


---

**Description**

Used within calls to pba and pbaAgain to obtain the probability a unit is assigned treatment given its fitted propensity score.

**Usage**

```
piFunction(fit, kparam, qparam)
```

**Arguments**

fit	Fitted propensity score.
kparam	Balancing parameter.
qparam	Global target for proportion of units treated.

**Details**

The input kparam must be one of 0, Inf, or the ratio of two positive odd integers. Both fit and qparam must be between 0 and 1.

**Value**

A numeric object. In the context of PBA, the probability of assignment to treatment for the current unit.

**Author(s)**

Travis M. Loux

**Examples**

```
piFunction(fit=0.6, kparam=1, qparam=0.5)
piFunction(fit=0.6, kparam=5, qparam=0.5)
piFunction(fit=0.6, kparam=1/5, qparam=0.5)
```

```
piFunction(fit=0.6, kparam=1, qparam=2/3)
piFunction(fit=0.6, kparam=5, qparam=2/3)
piFunction(fit=0.6, kparam=1/5, qparam=2/3)
```

---

plotpi

*Plots of piFunction*

---

**Description**

Can be used to investigate the strength of balance forced by various values of the tuning parameter  $k$ .

**Usage**

```
plotpi(k, global = 0.5)
addpi(k, global = 0.5, ...)
```

**Arguments**

<code>k</code>	Balancing parameter.
<code>global</code>	Global target for proportion of units treated.
<code>...</code>	Parameters in <code>addpi</code> passed to <code>lines</code> function.

**Details**

The function `plotpi` creates a plot of treatment probability against fitted propensity score for values of  $k$  and `global`. The function `addpi` adds a curve for a new combination of  $k$  and `global` to an existing plot.

**Author(s)**

Travis M. Loux

**Examples**

```
plotpi(k=3, global=0.5)  
addpi(k=5/3, lty=2, col='red')  
plotpi(k=1/3, global=2/3)
```



# Index

\* **design**

genPerms, 2

pba, 3

pcr, 5

\* **multivariate**

pba, 3

pcr, 5

addpi (plotpi), 7

genPerms, 2, 5

getVar, 3, 5

pba, 3

pbaAgain (pba), 3

pcr, 5

piFunction, 4, 6

plotpi, 7