# Package 'prefeR'

October 14, 2022

**Type** Package

**Title** R Package for Pairwise Preference Elicitation

**Version** 0.1.3

**Date** 2022-04-24

**Author** John Lepird

**Maintainer** John Lepird <jlepird@alum.mit.edu>

**Description** Allows users to derive multi-objective weights from pairwise comparisons, which research shows is more repeatable, transparent, and intuitive other techniques. These weights can be rank existing alternatives or to define a multi-objective utility function for optimization.

**License** MIT + file LICENSE

**Imports** mcmc, methods, entropy

**Suggests** testthat, knitr, rmarkdown

**VignetteBuilder** knitr

**RoxygenNote** 7.1.2

**Encoding** UTF-8

**URL** https://github.com/jlepird/prefeR, https://jlepird.github.io/prefeR/

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-04-24 23:00:02 UTC

## R topics documented:

---

## `.calculateLogProb` *Calculates the log probability of seeing a given set of preferences*

---

### Description

Calculates the log probability of seeing a given set of preferences

### Usage

```
.calculateLogProb(x, p)
```

### Arguments

| | |
|---|---|
| `x` | A guess for our weight vector |
| `p` | An object of the Bayes preference class |

### Value

A scalar log-likelihood of the guess x

---

## `.estimateEntropy` *Calculates the expected posterior entropy of the prefel object if x and y are compared. Ignores the odds of indifference preferences, as using them would increase runtime 50% without much gain.*

---

### Description

Calculates the expected posterior entropy of the prefel object if x and y are compared. Ignores the odds of indifference preferences, as using them would increase runtime 50% without much gain.

### Usage

```
.estimateEntropy(p, currentGuess, x, y)
```

### Arguments

|  |  |
|---|---|
| `p` | An object of class BayesPrefClass. |
| `currentGuess` | The current best estimate for our weight vector. |
| `x` | Possible comparison 1 |
| `y` | Possible comparison 2 |

---

## `.getLogIndifProb` *Evaluates the likelihood of the observed indifference preferences*

---

### Description

Evaluates the likelihood of the observed indifference preferences

### Usage

```
.getLogIndifProb(x, pref, p)
```

### Arguments

|  |  |
|---|---|
| `x` | the underlying data |
| `pref` | the stated preference |
| `p` | the preference elication object |

---

## `.getLogStrictProb` *Evaluates the likelihood of the observed strict preferences*

---

### Description

Evaluates the likelihood of the observed strict preferences

### Usage

```
.getLogStrictProb(x, pref, p)
```

### Arguments

|  |  |
|---|---|
| `x` | the underlying data |
| `pref` | the stated preference |
| `p` | the preference elication object |

## `.sampleEntropy` *Calculates the entropy of a matrix of samples.*

### Description

Calculates the entropy of a matrix of samples.

### Usage

```
.sampleEntropy(X)
```

### Arguments

| | |
|---|---|
| `X` | a matrix where each row is a sample of variables in different columns |

## `BayesPrefClass` *An object containing all data necessary for preference elicitation.*

### Description

An object containing all data necessary for preference elicitation.

### Fields

`data` A matrix or dataframe of data.

`priors` A list of functions that give the prior on each variable.

`sigma` A scalar value to use for the confusion factor (default 0.1).

`Sigma` (Internal use only) A matrix of sigma * diag(ncol(data)).

`strict` A list of lists of preferences. For each element x, x[[1]] > x[[2]].

`indif` A list of lists of indifference preferences. For each element x, x[[1]] = x[[2]].

`weights` A vector of weights determined by the inference algorithm.

### Methods

`addPref(x)` Adds a preference created using %>%, %<%, or %=%.

`infer(estimate = "recommended")` Calls the “infer” function to guess weights

`rank()` Calculates the utility of each row in our dataset

`suggest(maxComparisons = 10)` Calls the “suggest” function to guess weights

---

`Exp` *A convenience function for generating Exponential priors.*

---

### Description

A convenience function for generating Exponential priors.

### Usage

```
Exp(mu = 1)
```

### Arguments

| | |
|---|---|
| `mu` | The mean of the exponential distribution, i.e. $1/rate$ |

### Value

A function yielding the log-PDF at x of a exponential distribution with given statistics.

### See Also

Other priors: `Flat()`, `Normal()`

### Examples

```
Exp(1)(1) == dexp(1,1, log = TRUE)
```

---

`Flat` *A convenience function for generating a flat prior.*

---

### Description

A convenience function for generating a flat prior.

### Usage

```
Flat()
```

### Value

The zero function.

### See Also

Other priors: `Exp()`, `Normal()`

### Examples

```
Flat()(1) == 0.0
```

## infer — *A function that estimates the user's underlying utility function.*

### Description

A function that estimates the user's underlying utility function.

### Usage

```
infer(p, estimate = "recommended", nbatch = 1000)
```

### Arguments

| | |
|---|---|
| `p` | A BayesPrefClass instance. |
| `estimate` | The type of posterior point-estimate returned. Valid options are "recommended" (default), "MAP", and "mean". |
| `nbatch` | If using Monte Carlo estimates, the number of samples. Defaults to 1000. |

### Value

A vector of parameters that best fits the observed preferences.

### Examples

```
p <- prefEl(data = data.frame(c(1,0,1), c(0,1,1), c(1,1,1)),
            priors = c(Normal(0, 1), Exp(0.5), Flat()))
p$addPref(1 %>% 2)
infer(p, estimate = "MAP")
```

## Normal — *A convenience function for generating Normal priors.*

### Description

A convenience function for generating Normal priors.

### Usage

```
Normal(mu = 0, sigma = 1)
```

### Arguments

| | |
|---|---|
| `mu` | The mean of the normal distribution |
| `sigma` | The standard deviation of the prior |

## Value

A function yielding the log-PDF at x of a normal distribution with given statistics.

## See Also

Other priors: `Exp()`, `Flat()`

## Examples

```
Normal(0, 1)(1) == dnorm(1, log = TRUE)
```

---

`prefEl` *A shortcut to create objects of the class BayesPrefClass.*

---

## Description

A shortcut to create objects of the class BayesPrefClass.

## Usage

```
prefEl(data = NA, priors = list(), ...)
```

## Arguments

| | |
|---|---|
| `data` | A matrix or dataframe of data. Each column should be a variable, each row an observation. |
| `priors` | A list of functions that give the prior on each variable. E.g. see help(Flat) |
| `...` | Other parameters to pass to the class constructor. Not recommended. |

## Examples

```
p <- prefEl(data = data.frame(x = c(1,0,1), y = c(0, 1, 1)),
            priors = c(Normal(0,1), Flat()))
```

## `suggest` *Suggests a good comparison for the user to make next.*

### Description

Suggests a good comparison for the user to make next.

### Usage

```
suggest(p, maxComparisons = 10)
```

### Arguments

| | |
|---|---|
| `p` | An object of class BayesPrefClass. |
| `maxComparisons` | The maximum number of possible comparisons to check. Default: 10. |

### Value

A two-element vector of recommended comparisons.

## `%=%` *A helper function to add in preferences in a user-friendly way.*

### Description

A helper function to add in preferences in a user-friendly way.

### Usage

```
a %=% b
```

### Arguments

| | |
|---|---|
| `a` | The first alternative |
| `b` | The second alternative |

### See Also

Other preferences: `%<%()`, `%>%()`

### Examples

```
1 %=% 2 # indifferent between 1 and 2
```

## `%>%` *A helper function to add in preferences in a user-friendly way.*

### Description

A helper function to add in preferences in a user-friendly way.

### Usage

```
a %>% b
```

### Arguments

| | |
|---|---|
| `a` | The preferred row |
| `b` | The non-preferred row |

### See Also

Other preferences: `%<%()`, `%=%()`

### Examples

```
1 %>% 2 # prefer row 1 to row 2
```

## `%<%` *A helper function to add in preferences in a user-friendly way.*

### Description

A helper function to add in preferences in a user-friendly way.

### Usage

```
a %<% b
```

### Arguments

| | |
|---|---|
| `a` | The non-preferred row |
| `b` | The preferred row |

### See Also

Other preferences: `%=%()`, `%>%()`

### Examples

```
1 %<% 2 # prefer row 2 to row 1
```

# Index