Package 'markoffchains'

June 7, 2023

Title COMPUTATIONALLY DESCRIBING MARKOV CHAINS

Version 0.0.0.9000

Description
Simple functions for computing properties of discrete time space, finite state space Markov chains.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3.9000

R topics documented:

`absorbing_states` *Identify absorbing states of a Markov chain*

Description

This function takes the transition probability matrix as input and finds absorbent states.

Usage

```
absorbing_states(P, mode = "binarr")
```

Arguments

P	The transition probability matrix.
mode	Can take value "binarr" (producing a binary array) or "indices" (returning the absorbent state indices).

Value

A binary array (if mode == "binarr") or an array of indices (if mode == "indices").

Examples

```
mtx = diag(1, 3, 3)
absorbing_states(mtx, "binarr")
absorbing_states(mtx, "indices")
```

---

access *This function takes the transition probability matrix as input and finds states accessible from each state.*

Description

This function takes the transition probability matrix as input and finds states accessible from each state.

Usage

```
access(P)
```

Arguments

P	The transition probability matrix.

Value

A matrix of binary values where the i, j th entry indicates whether the jth state is accessible from the ith one.

Examples

```
mtx = diag(1, 3, 3)
access(mtx)

mtx2 = matrix(1/3, 3, 3)
access(mtx2)
```

classify_classes *Classifies equivalent classes as recurrent and transient.*

Description

This function takes the transition probability matrix as input and returns the classification of equivalent classes.

Usage

```
classify_classes(P, mode = "bin_arr")
```

Arguments

P	The transition probability matrix.
mode	"binarr" (returns binary vector with ith entry indicating whether the ith class is recurrent) or "visual" (returns readable dataframe with class elements and classification)

Value

A binary array or dataframe

Examples

```
mtx = diag(1, 3, 3)
classify_classes(mtx, "visual")

mtx2 <- matrix(1/3, 3, 3)
classify_classes(mtx2)
```

classify_states *Function to classify states as recurrent or transient*

Description

This function takes the transition probability matrix as input and returns whether it is recurrent (1) or transient (0)

Usage

```
classify_states(P, mode = "bin_arr")
```

Arguments

P	The transition probability matrix.
mode	Ouptut format: Either "binarr" or "visual"

Value

A matrix of binary values or dataframe

Examples

```
mtx <- matrix(c(0, 1, 0, 0, 0, 1, 0, 1, 0 ),3, 3, byrow =TRUE)
classify_states(mtx, "visual")

mtx2 <- matrix(1/3, 3, 3)
classify_states(mtx2)
```

---

comm_classes *Finds communicating classes in a stochastic process.*

Description

This function takes the transition probability matrix as input and returns the communicating classes.

Usage

```
comm_classes(P, mode = "bin_arr")
```

Arguments

P	The transition probability matrix.
mode	"binarr" (returns binary array with i,j th entry indicating whether jth element is present in ith class) or "list" (with ith element of list being a vector of states present in ith class)

Value

A binary array or list of vectors of

Examples

```
mtx = diag(1, 3, 3)
comm_classes(mtx, "list")

mtx2 <- matrix(1/3, 3, 3)
comm_classes(mtx2)
```

is_stochastic *Checks whether a matrix is stochastic.*

Description

This function takes any matrix as input and finds whether it is a valid transition probability matrix of a stochastic process.

Usage

```
is_stochastic(P)
```

Arguments

P	The matrix.

Value

A boolean value representing whether it is stochastic.

Examples

```
mtx = diag(1, 3, 3)
is_stochastic(mtx)

mtx2 <- matrix(1, 2, 3)
is_stochastic(mtx2)
```

mem_onehot *Helper function to create concatenated one hot vectorss*

Description

Helper function to create concatenated one hot vectorss

Usage

```
mem_onehot(idx, n)
```

Arguments

idx	A numeric matrix containing the indices to convert to one hot
n	The number of states in the MC

Value

A matrix with concatenated one hot vectors

Examples

```
idx <- c(2,3)
mem_onehot(idx, 6)
```

`reachable` *Checks whether a state is reachable from another state.*

Description

This function takes the transition probability matrix as input and finds whether the jth state is accessible/reachable from the ith state.

Usage

```
reachable(P, i, j)
```

Arguments

`P`	The transition probability matrix.
`i`	The original state.
`j`	The target state.

Value

A boolean value representing whether it is reachable.

Examples

```
mtx = diag(1, 3, 3)
reachable(mtx, 1, 2)

mtx2 <- matrix(1/3, 3, 3)
reachable(mtx2, 1, 2)
```

`steady_class_probs` *Function to find steady state probabilities of entering into different communicating classes*

Description

This function takes the transition probability matrix as input and returns probabilities of ending up in each class from each state

Usage

```
steady_class_probs(P)
```

Arguments

P	The transition probability matrix.

Value

A matrix (n_states * n_classes) of probabilities

Examples

```
test <- matrix(c(0, 0.25, 0.5, 0.25, 0, 0,
0, 0.5, 0, 0.5, 0, 0,
0, 0, 0.25, 0, 0.75, 0,
0, 0.5, 0, 0.5, 0, 0,
0, 0, 0.75, 0, 0.25, 0,
0, 0.25, 0, 0.25, 0.5, 0
) , 6, 6, byrow = TRUE)

steady_class_probs(test)
```

---

steady_recurrent *Outputs stable configuration for a recurrent Markov chain*

Description

This function takes the transition probability matrix for a recurrent Markov chain as input and finds absorbent states.

Usage

```
steady_recurrent(P)
```

Arguments

P	The transition probability matrix.

Value

The probability matrix of the states after t= infinity.

Examples

```
mtx = matrix(0.25, 4, 4)
steady_recurrent(mtx)
```

steady_return_probs *Function to find steady state probabilities of returning to the initial state*

Description

This function takes the transition probability matrix as input and returns probabilities of returning to the same state at t = infinity.

Usage

```
steady_return_probs(P)
```

Arguments

P	The transition probability matrix.

Value

A matrix (1 * n_states) of probabilities

Examples

```
test <- matrix(c(0, 0.25, 0.5, 0.25, 0, 0,
0, 0.5, 0, 0.5, 0, 0,
0, 0, 0.25, 0, 0.75, 0,
0, 0.5, 0, 0.5, 0, 0,
0, 0, 0.75, 0, 0.25, 0,
0, 0.25, 0, 0.25, 0.5, 0
) , 6, 6, byrow = TRUE)

steady_return_probs(test)
```

steady_state *Function to find stationary matrix (probabilities in steady state)*

Description

This function takes the transition probability matrix as input and returns the probability matrix of ending up at each state at t = infinity.

Usage

```
steady_state(P)
```

Arguments

P	The transition probability matrix.

Value

A stochastic matrix (n_states * n_states) of probabilities

Examples

```
test <- matrix(c(0, 0.25, 0.5, 0.25, 0, 0,
0, 0.5, 0, 0.5, 0, 0,
0, 0, 0.25, 0, 0.75, 0,
0, 0.5, 0, 0.5, 0, 0,
0, 0, 0.75, 0, 0.25, 0,
0, 0.25, 0, 0.25, 0.5, 0
) , 6, 6, byrow = TRUE)

steady_state(test)
```

---

submtx *Helper function to create a submatrix of a TPM consisting of specific states*

Description

Helper function to create a submatrix of a TPM consisting of specific states

Usage

```
submtx(P, idx)
```

Arguments

P	The TPM
idx	A numeric matrix containing the indices

Value

A matrix with only the specific states (generally for a recurrent class)

Examples

```
idx <- c(2,3)
P <- matrix(c(1,0,0, 0, 1, 0, 0, 0, 1), 3, 3, byrow =TRUE)
submtx(P, idx)
```

Index