```
function q=ReinforcementLearning_RandomPol(R, gamma, goalState, alpha)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Original Q Learning by Example code, by Kardi Teknomo
% (http://people.revoledu.com/kardi/)
%
% Code amended by Ioannis Makris and Andrew Chalikiopoulos
% Model for an agent to find shortest path through a 5x5 maze grid
% This algorithm uses a random policy to choose the next state
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
format short
format compact

% Three inputs: R, gamma and alpha
if nargin<1,
% immediate reward matrix
    R=RewardMatrix25;
end
if nargin<2,
    gamma=0.80;              % discount factor
    alpha=0.80;              % learning rate
end
if nargin<3
    goalState=22;
end

q=zeros(size(R));         % initialize Q as zero
q1=ones(size(R))*inf;     % initialize previous Q as big number
count=0;                  % counter
steps=0;                  % counts the number of steps to goal
episodes=0;               % counts the number of episodes
B=[];                     % matrix to add results of steps and episode count
cumReward=0;              % counter to calculate accumulated reward



for episode=0:50000 % the amount of episodes to run

    state=5;        % Starting state of the agent


    while state~=goalState            % loop until find goal state
        % select any action from this state
        x=find(R(state,:)>=0)         % find possible action of this state
        if size(x,1)>0,
            x1=RandomPermutation(x);   % randomize the possible action
            x1=x1(1);                  % select an action (only the first element
            cumReward=cumReward+q(state,x1);
        end

        x2 = find(R(x1,:)>=0);   % find possible steps from next step
        qMax=(max(q(x1,x2(1:end)))); % extract qmax from all possible next states
```

```
            q(state,x1)= q(state,x1)+alpha*((R(state,x1)+gamma*qMax)-q(state,x1))    %
            state=x1;    % set state to next state

            if state~=goalState     % keep track of steps taken if goal not reached
                steps=steps+1;

            else
                episodes=episodes+1; % if goal reach increase episode counter
                A=[episodes; steps; cumReward;];   % create episodes, steps and cumRew
                B=horzcat(B, A);    % add the new results to combined matrix
                steps=0;    % reset steps counter to 0
                cumReward=0;    % reset cumReward counter to

            end

        end


         % break if convergence: small deviation on q for 1000 consecutive
         if sum(sum(abs(q1-q)))<0.00001 && sum(sum(q >0))
             if count>1000,
                 episode  % report last episode
                 break % for loop
             else
                 count=count+1; % set counter if deviation of q is small
             end
         else
             q1=q
             count=0;  % reset counter when deviation of q from previous q is large
         end
    end

    % row 4 in matrix is cumReward/steps taken per episode
    B(4,:) = (B(3,:)./B(2,:));

    %episodes vs cumReward taken averaged against steps taken
    %plot(B(1,:),B(4,:));

    % create a plot of episodes vs steps taken and episodes vs cumReward taken average
    figure % new figure
    [hAx] = plotyy(B(1,1 : 5 : end),B(2,1 : 5 : end), B(1,1 : 5 : end),B(4,1 : 5 : end

    title('Q-Learning Performance')
    xlabel('Episodes')
    ylabel(hAx(1),'Steps') % left y-axis
    ylabel(hAx(2),'Cumulative Reward/Steps') % right y-axis

    % create a plot of episodes vs cumReward
    %plot(B(1,1 : 5 : end),B(3,1 : 5 : end));

    %normalize q
    g=max(max(q));
    if g>0,
        q=100*q/g;
```

```
end

% display the shortest path to the goal
  Optimal=[];
  state=5;
  Optimal=horzcat(Optimal,state);

  while state~=goalState

            [~,optimal]=(max(q(state,:)));
            state = optimal;
            Optimal=horzcat(Optimal,state);
  end
  display('Shortest path:')
  display(Optimal);
```

Published with MATLAB® R2014a