If only R is desired, then houseqr does the job. In order to obtain R, we need to compose the Householder transformations. We present a simple method which is not the most efficient (there is a way to avoid multiplying explicity the Householder matrices).

The function buildhouse creates a Householder reflection from a vector v.

```
function P = buildhouse(v,i)
% This function builds a Householder reflection
%   [I 0 ]
%   [0 PP]
%   from a Householder reflection
%   PP = I - 2uu*uu'
%   where uu = v(i:n)
%   If uu = 0 then P - I
%

n = size(v,1);
if v(i:n) == zeros(n - i + 1,1)
   P = eye(n);
else
   PP = eye(n - i + 1) - 2*v(i:n)*v(i:n)';
   P = [eye(i-1) zeros(i-1, n - i + 1); zeros(n - i + 1, i - 1) PP];
end
end
```

The function buildQ builds the matrix Q in the QR-decomposition of A.

```
function Q = buildQ(u)
% Builds the matrix Q in the QR decomposition
% of an nxn  matrix A using Householder matrices,
% where u is a representation of the n - 1
% Householder reflection by a list u of vectors produced by
% houseqr

n = size(u,1);
Q = buildhouse(u(:,1),1);
for i = 2:n-1
  Q = Q*buildhouse(u(:,i),i);
end
end
```

The function buildhouseQR computes a QR-factorization of A. At the end, if some entries on the diagonal of R are negative, it creates a diagonal orthogonal matrix P such that PR has nonnegative diagonal entries, so that $A = (QP)(PR)$ is the desired QR-factorization of A.