Markov Chain

1. Definition

In [1]:

```
A = [ 0.9 0.15 0.25 ; 0.075 0.8 0.25 ; 0.025 0.05 0.5 ];
x_1 = [0,1,0];
B1 = A*x_1;
println("At time n+1, the distribution is:","[",B1',"]")
```

```
At time n+1, the distribution is:[[0.15 0.8 0.05]]
```

In line 1 we construct a 3*3 matrix.

In line 2 we set a stochastic row vector x_1.

The distribution over states is line 3 which A multiplied by x_1.

In [2]:

```
A = [ 0.9 0.15 0.25 ; 0.075 0.8 0.25 ; 0.025 0.05 0.5 ];
x_1 = [0,1,0];
B2 = A*B1;
println("At time n+2, the distribution is:","[",B2',"]")
```

```
At time n+2, the distribution is:[[0.2675 0.66375 0.06875]]
```

Similar function with before, calculate the distribution at time n+2

In [3]:

```
A = [ 0.9 0.15 0.25 ; 0.075 0.8 0.25 ; 0.025 0.05 0.5 ];
x_1 = [0,1,0];
B3 = A*B2;
println("At time n+3, the distribution is:","[",B3',"]")
```

```
At time n+3, the distribution is:[[0.3575 0.56825 0.07425]]
```

Similar function with before, calculate the distribution at time n+3

In [4]:

```
A = [ 0.9 0.15 0.25 ; 0.075 0.8 0.25 ; 0.025 0.05 0.5 ];
x_1 = [0,1,0];
B4 = A*B3;
println("At time n+4, the distribution is:","[",B4',"]")
```

At time n+4, the distribution is:[[0.42555 0.499975 0.074475]]

Similar function with before, calculate the distribution at time n+4

2. Examples of Markov chains

In [5]:

```
T = 210 ;
A = [ 0.9 0.15 0.25 ; 0.075 0.8 0.25 ; 0.025 0.05 0.5 ];
x_1 = [0,1,0];
state_traj = [x_1 zeros(3,T-1) ]; # State trajectory
for t=1:T-1 # Dynamics recursion
    state_traj[:,t+1] = A*state_traj[:,t];
end
using Plots
plot(1:T, state_traj', xlabel = "Time t",
              label = ["Bull Market", "Bear Market", "Stagnant Market"])
```

Out[5]:

0 50 100 150 200 0.00 0.25 0.50 0.75 1.00 Time t Bull Market Bear Market Stagnant Market

Demonstration 1:

In line 1 we set the calculate time as 210.

In line 2 we construct a 3 *3 matrix.*

In line 3 we set a stochastic row vector x_1.

In line 4 we conbine the x_1 vector and zero matrix to be a new matrix state_traj.

In line 5~7, we use dynamics recursion to update the state_traj matrix's value by *A*state_traj.

In line 8~9, we import and use plot function to draw markov chain prediction charm.

In [10]:

```
T = 210 ;
A = [ 0.9 0.15 0.25 ; 0.075 0.8 0.25 ; 0.025 0.05 0.5 ];
x_2 = [0.3,0.4,0.3];
state_traj = [x_2 zeros(3,T-1) ]; # State trajectory
for t=1:T-1 # Dynamics recursion
    state_traj[:,t+1] = A*state_traj[:,t];
end
using Plots
plot(1:T, state_traj', xlabel = "Time t",
              label = ["Bull Market", "Bear Market", "Stagnant Market"])
```

Out[10]:

0 50 100 150 200 0.1 0.2 0.3 0.4 0.5 0.6 Time t Bull Market Bear Market Stagnant Market

Demonstration 2:

In line 1 we set the calculate time as 210.

In line 2 we construct a 3 * 3 matrix.

In line 3 we set a stochastic row vector x_2.

In line 4 we conbine the x_2 vector and zero matrix to be a new matrix state_traj.

In line 5~7, we use dynamics recursion to update the state_traj matrix's value by Astate_traj.

In line 8~9, we import and use plot function to draw markov chain prediction charm.

In [11]:

```
T = 210 ;
A = [ 0.9 0.15 0.25 ; 0.075 0.8 0.25 ; 0.025 0.05 0.5 ];
x_3 = [0.5,0.2,0.1];
state_traj = [x_3 zeros(3,T-1) ]; # State trajectory
for t=1:T-1 # Dynamics recursion
    state_traj[:,t+1] = A*state_traj[:,t];
end
using Plots
plot(1:T, state_traj', xlabel = "Time t",
            label = ["Bull Market", "Bear Market", "Stagnant Market"])
```

Out[11]:

0 50 100 150 200 0.1 0.2 0.3 0.4 0.5 Time t Bull Market Bear Market Stagnant Market

Demonstration 3: In line 1 we set the calculate time as 210.

In line 2 we construct a 3 * 3 matrix.

In line 3 we set a stochastic row vector x_3.

In line 4 we conbine the x_3 vector and zero matrix to be a new matrix state_traj.

In line 5~7, we use dynamics recursion to update the state_traj matrix's value by Astate_traj.

In line 8~9, we import and use plot function to draw markov chain prediction charm.