Example 6.1. Two forces, which we will call red (R) and blue (B), are engaged in battle. In this conventional battle, attrition is due to direct fire (infantry) and area fire (artillery). The attrition rate due to direct fire is assumed proportional to the number of enemy infantry. The attrition rate due to artillery depends on both the amount of enemy artillery and the density of friendly troops. Red has amassed five divisions to attack a blue force of two divisions. Blue has the advantage of defense, and superior weapon effectiveness besides. How much more effective does blue have to be in order to prevail in battle?

In [1]:
```
lambda=1.0
a2=0.05
b2=0.005
a1=lambda*a2
b1=lambda*b2
```

Out[1]:
```
0.005
```

In [2]:
```
R0=5.0
B0=2.0
```

Out[2]:
```
2.0
```

In [3]:
```
dR(R,B)=-a1*B-b1*R*B
dB(R,B)=-a2*R-b2*R*B
```

Out[3]:
```
dB (generic function with 1 method)
```

In [4]:
```
Rn=R0
Bn=B0
for n=2:10
    Rn=Rn+dR(Rn,Bn)
    Bn=Bn+dB(Rn,Bn)
    println("n=$n R=$Rn B=$Bn")
end
```

```
n=2 R=4.85 B=1.709
n=3 R=4.7231067499999995 B=1.43248571532125
n=4 R=4.617653549477375 B=1.1685294241077373
n=5 R=4.53224775805749 B=0.9154367118916803
n=6 R=4.465730992537333 B=0.6717096917848079
n=7 R=4.417147134005137 B=0.43601713238479733
n=8 R=4.385716518252444 B=0.20717006877346977
n=9 R=4.370815068850234 B=-0.015898194961091072
n=10 R=4.371957418948806 B=-0.23414853475149122
```

Good to visualize what is happening when modeling a problem

In [5]:
```
Rn=R0; Rs=[R0]
Bn=B0; Bs=[B0]
for n=2:10
    Rn=Rn+dR(Rn,Bn); push!(Rs,Rn)
```

```
    Bn=Bn+dB(Rn,Bn); push!(Bs,Bn)
end
```

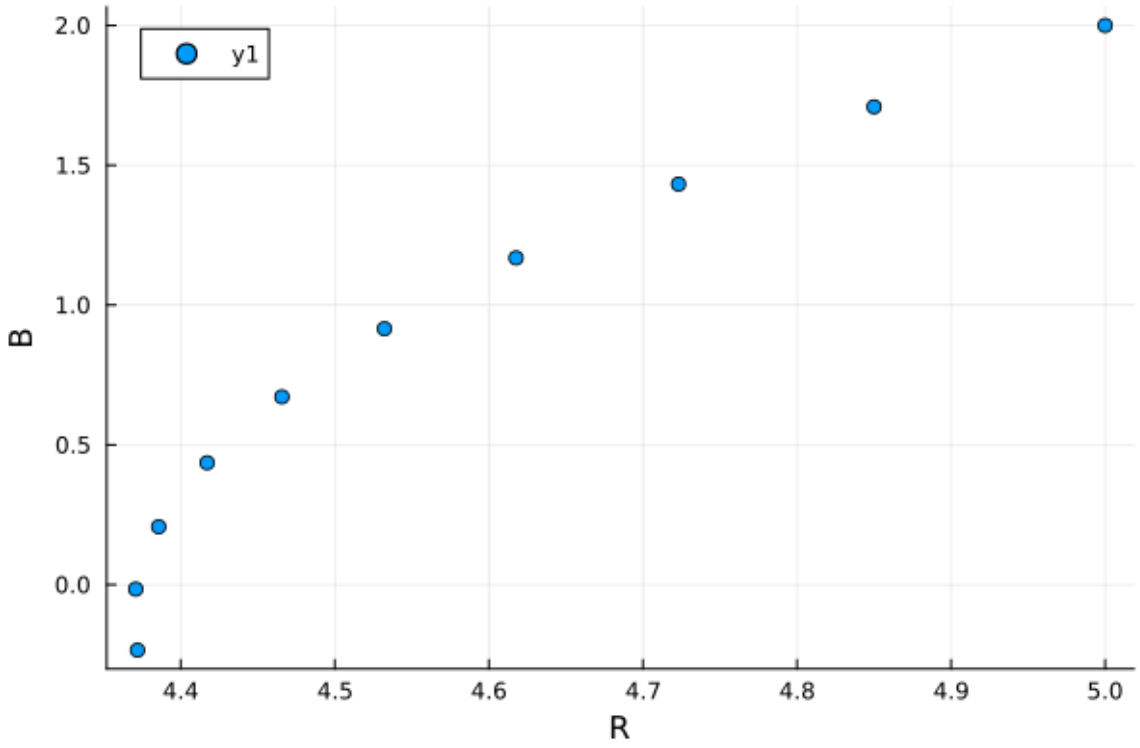
In [6]:
```
[Rs Bs]
```

Out[6]:
```
10×2 Matrix{Float64}:
 5.0       2.0
 4.85      1.709
 4.72311   1.43249
 4.61765   1.16853
 4.53225   0.915437
 4.46573   0.67171
 4.41715   0.436017
 4.38572   0.20717
 4.37082  -0.0158982
 4.37196  -0.234149
```

In [7]:
```
using Plots
```

In [8]:
```
scatter(Rs,Bs,xlabel="R",ylabel="B")
```

Out[8]:

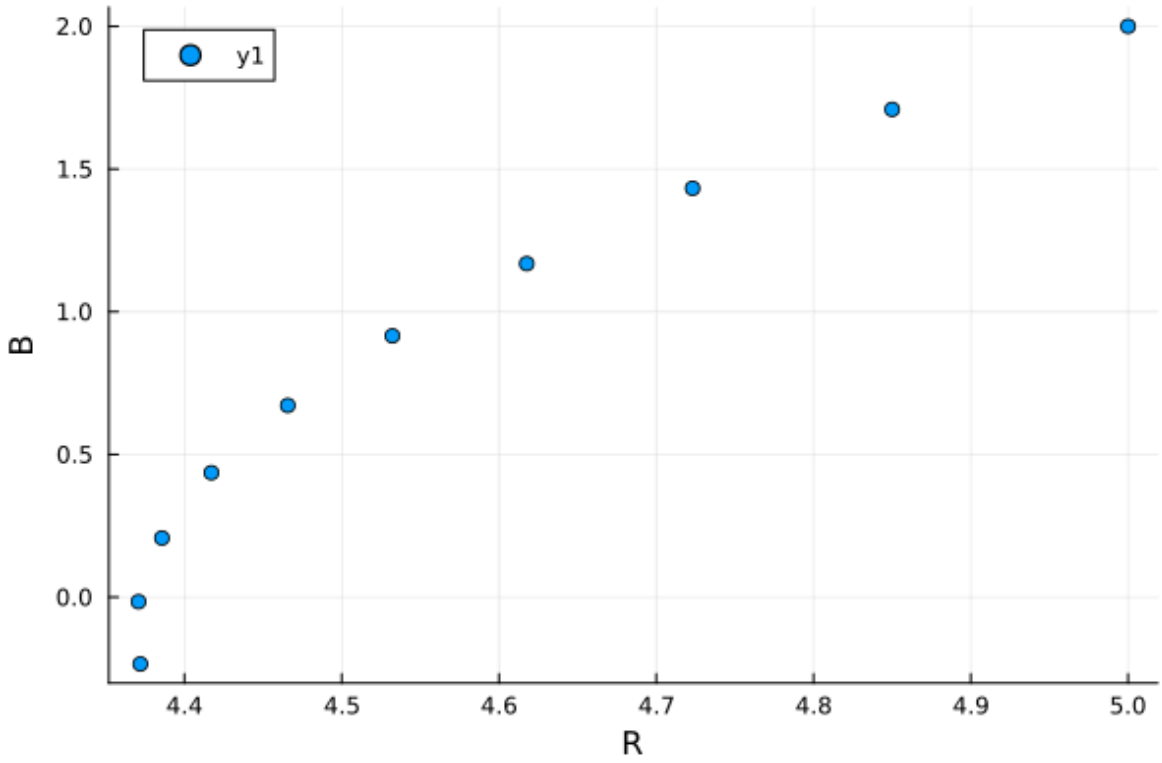


In [9]:
```
function battle(lambda,nmax)
    global a1=lambda*a2
    global b1=lambda*b2
    Rn=R0; Rs=[R0]
    Bn=B0; Bs=[B0]
    for n=2:nmax
        Rn=Rn+dR(Rn,Bn); push!(Rs,Rn)
        Bn=Bn+dB(Rn,Bn); push!(Bs,Bn)
    end
```

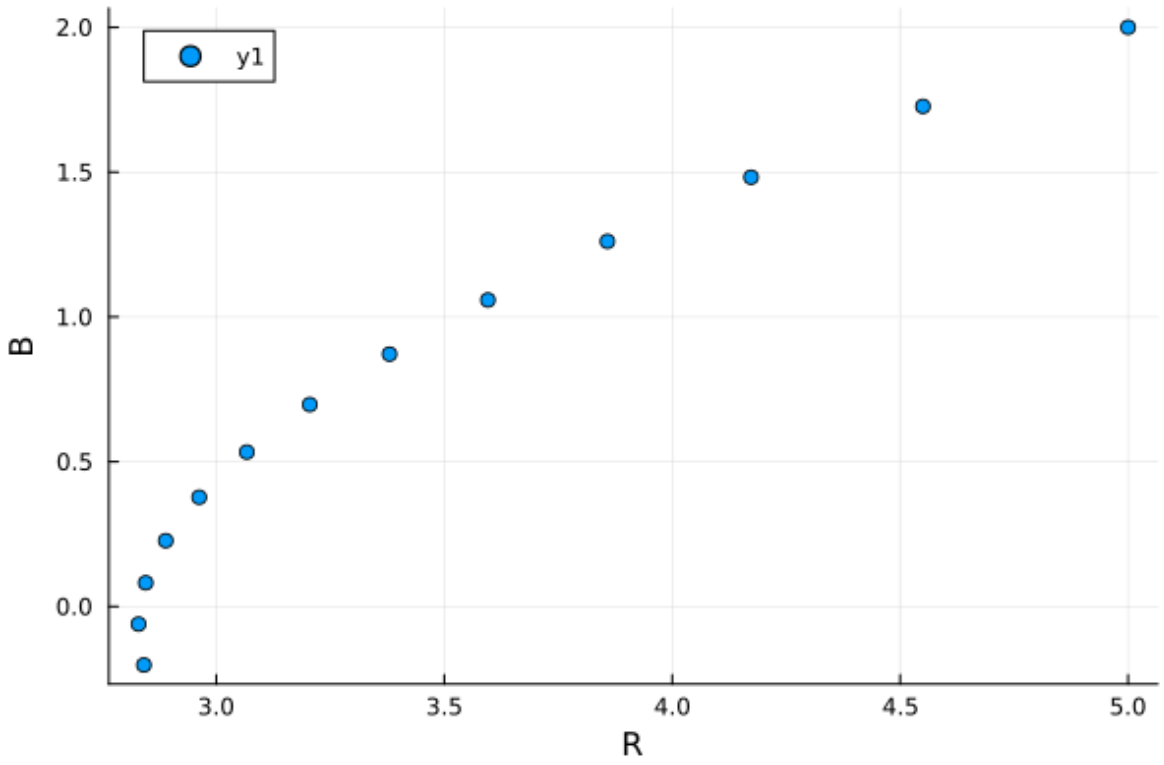
```
    display(scatter(Rs,Bs,xlabel="R",ylabel="B"))
end
```

Out[9]: battle (generic function with 1 method)

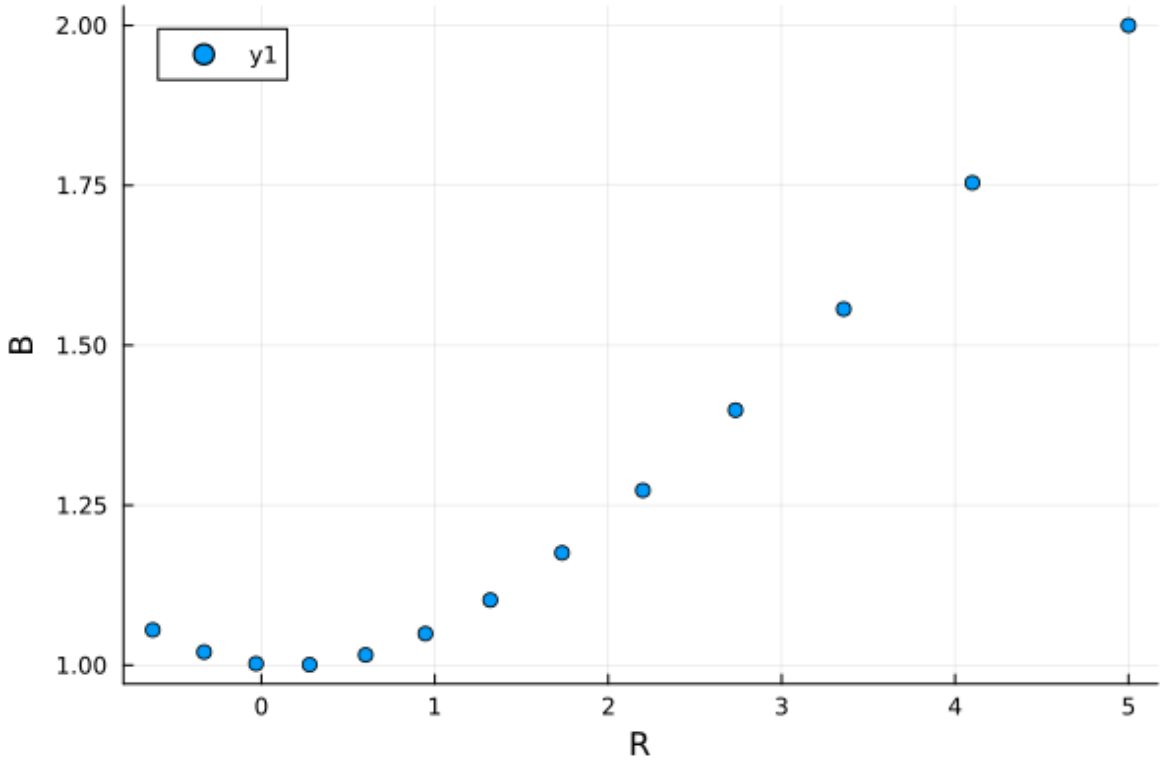
In [10]:
```
battle(1.0,10)
```



In [11]:
```
battle(3.0,13)
```



In [12]:
```
battle(6.0,13)
```



We decided to run the simulation for up to 14 days of combat (or N = 168 hours). The duration of battle is defined to be the number of hours of actual combat (there are 12 hours of fighting per day) until one of the variables x1 or x2

becomes zero or negative. If both sides survive 168 hours of combat, we call it a draw.

In [13]:
```
function winred(lambda,nmax)
    global a1=lambda*a2
    global b1=lambda*b2
    Rn=R0; Rs=[R0]
    Bn=B0; Bs=[B0]
    for n=2:nmax
        Rn=Rn+dR(Rn,Bn); push!(Rs,Rn)
        Bn=Bn+dB(Rn,Bn); push!(Bs,Bn)
        if Bn<=0
            return true
        end
        if Rn<=0
            return false
        end
    end
    return false
end
```

Out[13]:
```
winred (generic function with 1 method)
```

In [14]:
```
function winblue(lambda,nmax)
    global a1=lambda*a2
    global b1=lambda*b2
    Rn=R0; Rs=[R0]
    Bn=B0; Bs=[B0]
    for n=2:nmax
        Rn=Rn+dR(Rn,Bn); push!(Rs,Rn)
        Bn=Bn+dB(Rn,Bn); push!(Bs,Bn)
        if Rn<=0
            return true
        end
        if Bn<=0
            return false
        end
    end
    return false
end
```

Out[14]:
```
winblue (generic function with 1 method)
```

In [15]:
```
winblue(6.0,168)
```

Out[15]:
```
true
```

In [16]:
```
winred(6.0,168)
```

Out[16]:
```
false
```

In [17]:
```
winblue(3.0,168)
```

Out[17]: false

In [18]:
```
winred(3.0,168)
```

Out[18]: true

In [19]:
```
a=3.0
b=6.0
c=(a+b)/2
winred(c,168)
a=c
c=(a+b)/2
winred(c,168)
```

Out[19]: false

In [20]:
```
c
```

Out[20]: 5.25

In [21]:
```
winblue(c,168)
```

Out[21]: true

In [22]:
```
a=3.0
b=6.0
for l=1:10
    c=(a+b)/2
    println("checking [$a,$b] at c=$c...")
    if winred(c,168)
        println("lambda=",c," then red wins")
        a=c
    else
        println("lambda=",c," then red does not win")
        b=c
    end
end
c=(a+b)/2
```

```
checking [3.0,6.0] at c=4.5...
lambda=4.5 then red wins
checking [4.5,6.0] at c=5.25...
lambda=5.25 then red does not win
checking [4.5,5.25] at c=4.875...
lambda=4.875 then red does not win
checking [4.5,4.875] at c=4.6875...
lambda=4.6875 then red does not win
checking [4.5,4.6875] at c=4.59375...
lambda=4.59375 then red wins
checking [4.59375,4.6875] at c=4.640625...
lambda=4.640625 then red does not win
checking [4.59375,4.640625] at c=4.6171875...
lambda=4.6171875 then red wins
checking [4.6171875,4.640625] at c=4.62890625...
lambda=4.62890625 then red wins
checking [4.62890625,4.640625] at c=4.634765625...
lambda=4.634765625 then red wins
checking [4.634765625,4.640625] at c=4.6376953125...
lambda=4.6376953125 then red wins
```

Out[22]: 4.63916015625

In [23]:
```
a=3.0
b=6.0
for l=1:100
    c=(a+b)/2
    if winred(c,168)
        a=c
    else
        b=c
    end
end
c=(a+b)/2
```

Out[23]: 4.639637090131519

In [24]:
```
a=3.0
b=6.0
for l=1:100
    c=(a+b)/2
    if winblue(c,168)
        b=c
    else
        a=c
    end
end
c=(a+b)/2
```

Out[24]: 4.639637090131524

In [25]:
```
winblue(6,12)
```

Out[25]: true

In [26]:
```
winred(3,12)
```

Out[26]:
```
true
```

In [27]:
```
a=3.0
b=6.0
for l=1:100
    c=(a+b)/2
    if winred(c,12)
        a=c
    else
        b=c
    end
end
c=(a+b)/2
```

Out[27]:
```
3.1851982913899617
```

In [28]:
```
a=3.0
b=6.0
for l=1:100
    c=(a+b)/2
    if winblue(c,12)
        b=c
    else
        a=c
    end
end
c=(a+b)/2
```

Out[28]:
```
5.708719030676319
```

In [ ]:
```

```