```
using Pkg
```

```
Pkg.activate("C:/Users/PhytoGreg/.julia/pluto_notebooks")
```

```
begin
    using CSV
    using Plots
    using Dates
    using Distributions
    using Printf
    using LaTeXStrings
end
```

Load the ensemble from the .CSV

	trial	t	X1	X2	X3	X4
1	1	0.0	9999	0	1	0
2	1	1.0	9998	0	2	0
3	1	2.0	9998	0	2	0
4	1	3.0	9998	0	2	0
5	1	4.0	9997	1	2	0
6	1	5.0	9997	1	1	1
7	1	6.0	9997	1	1	1
8	1	7.0	9997	1	1	1
9	1	8.0	9997	0	2	1
10	1	9.0	9997	0	2	1
⋮	more					
15050	50	300.0	9997	0	0	3

```
begin
    Npop = 10000
    b    = 0.3
    dd = CSV.read(@sprintf "SEIR_ensemble_n=%.0f_b=%.2f.csv" Npop b)
end
```

Subset the ensemble for a single trial

d =

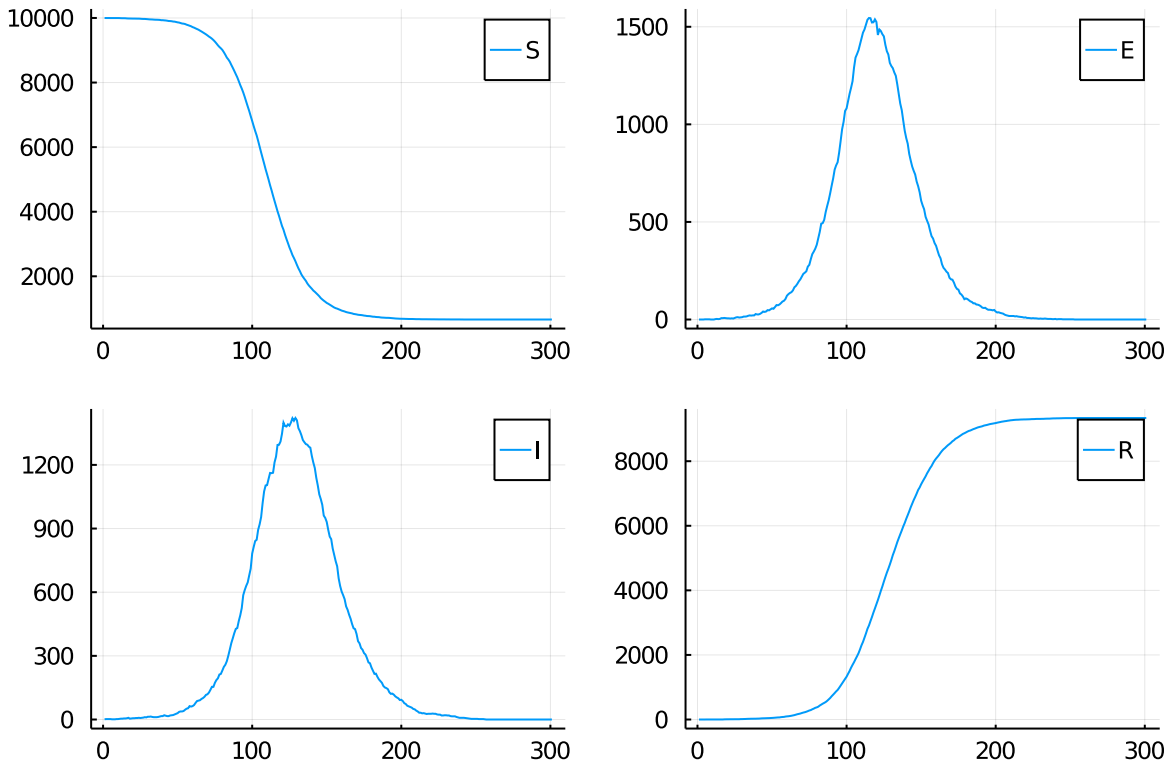
	trial	t	X1	X2	X3	X4
1	1	0.0	9999	0	1	0
2	1	1.0	9998	0	2	0
3	1	2.0	9998	0	2	0
4	1	3.0	9998	0	2	0
5	1	4.0	9997	1	2	0
6	1	5.0	9997	1	1	1
7	1	6.0	9997	1	1	1
8	1	7.0	9997	1	1	1
9	1	8.0	9997	0	2	1
10	1	9.0	9997	0	2	1
⋮	more					
201	1	300.0	662	0	0	9338

```
d = dd[dd.trial .==1,:]
```

```
▸ Int64[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5, 6, 7, ⋯ more ,
```

```
begin
    S = d.X1
    E = d.X2
    I = d.X3
    R = d.X4
end
```

Plot the trial




```
begin
    p1 = plot(S)
    p2 = plot(E)
    p3 = plot(I)
    p4 = plot(R)
    plot(p1,p2,p3,p4,labels=["S" "E" "I" "R"])
end
```

SIMULATE TESTING

Perfect random sampling

Sampling the population with the probability of a positive test equal to the fraction of infected individuals in the population, according to

$$p(t) = \frac{I(t)}{N_{pop}}.$$

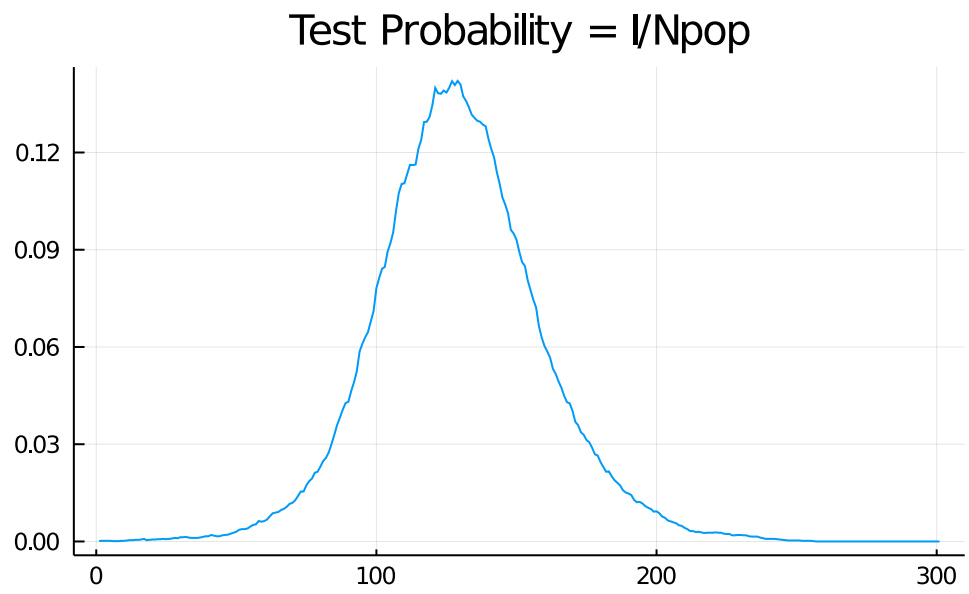
The simplest case is a constant number of tests performed each day. Below we will investigate time-dependent number of tests performed per day

```
▸ Int64[10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
```

```
###--CONSTANT NUMBER PER DAY--######
```

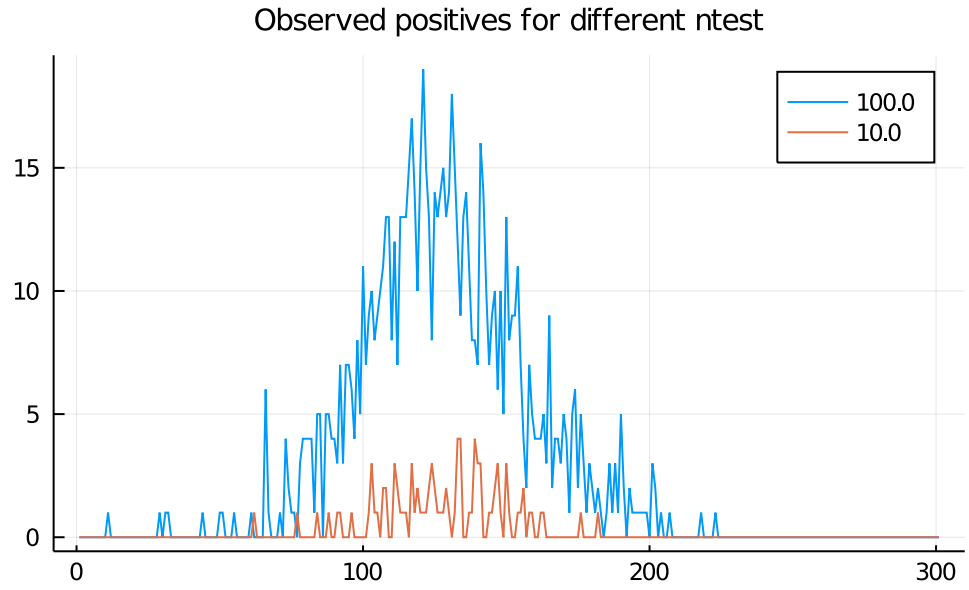
```
begin
    ptest1 = 0.01 #proportion of the population tested per day
    ptest2 = 0.001 #proportion of the population tested per day
    ntest1 = ptest1*Npop
    ntest2 = ptest2*Npop
    n      = length(S)
    Ntest1 = repeat([convert(Int64,round(ntest1,digits=0))],Int(n)) #time series of
    Ntest2 = repeat([convert(Int64,round(ntest2,digits=0))],Int(n)) #time series of
tests
end
```

Plot the time-dependent fraction of the infected individuals




```
begin
    Ifrac = I/Npop
    Ifrac[Ifrac.<=0.0] .= 0.0
    plot(Ifrac,size=(500,300),title="Test Probability = I/Npop",legend=false)
end
```

Plot a simulated time series for positive tests over time



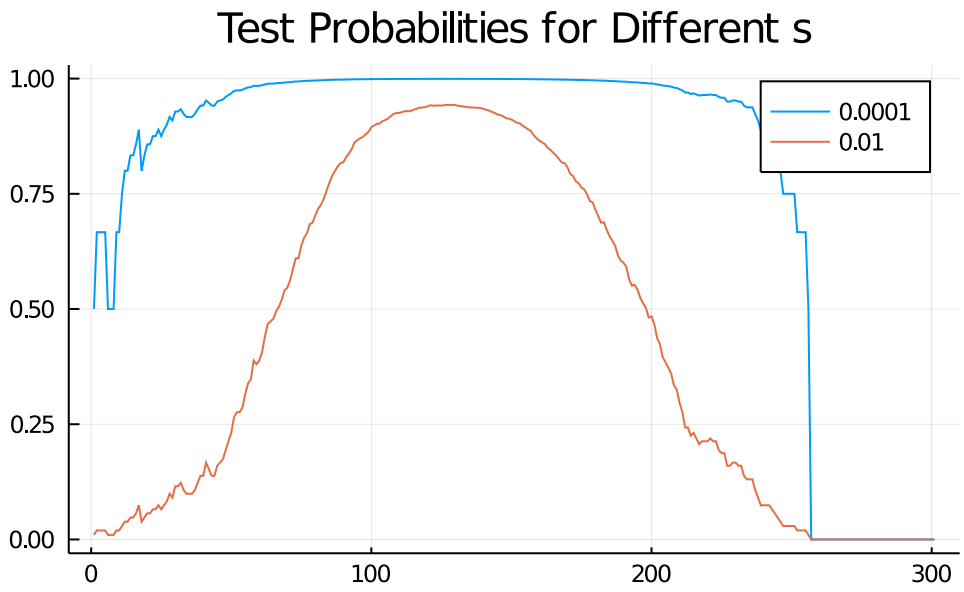

```
begin
    binom1 = Binomial.(Ntest1,Ifrac)
    binom2 = Binomial.(Ntest2,Ifrac)
    t1 = rand.(binom1)
    t2 = rand.(binom2)
    plot(t1,size=(500,300),labels=ntest1,titlefontsize=10, title = @sprintf
"Observed positives for different ntest")
    plot!(t2,size=(500,300),labels=ntest2)
end
```

Biased sampling

Here we introduce a tendency to sample infected individuals over others.

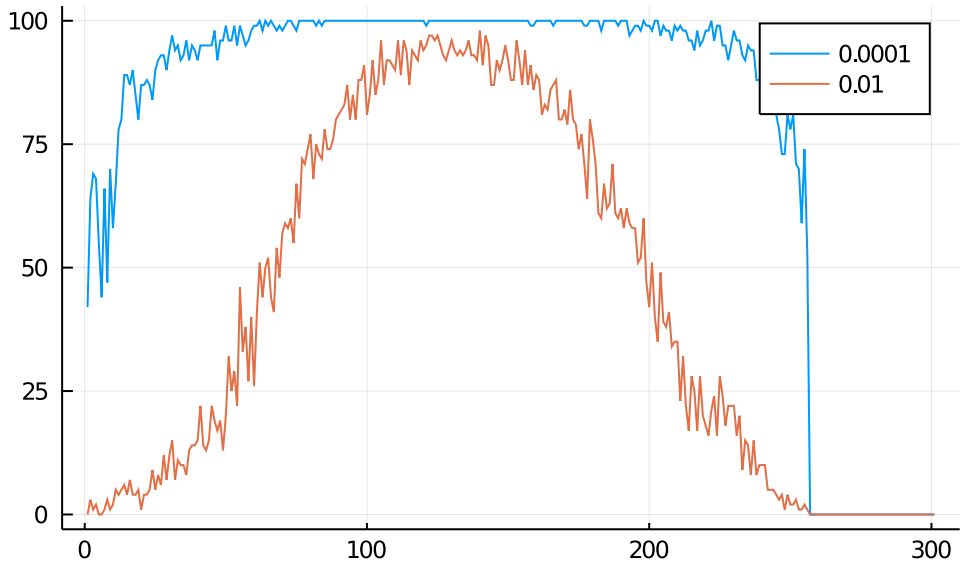
This is done by controlling the fraction of non-infected individuals that are available to testing, according to

$$p(t) = \frac{I(t)}{I(t) + s(N_{pop} - I(t))}.$$




```
begin
    s1 = 0.0001
    s2 = 0.01
    psamp1 = I./(I .+ s1.*(Npop.-I))
    psamp2 = I./(I .+ s2.*(Npop.-I))
    #psamp[psamp.<=0.0] .= 0.0
    plot(psamp1,size=(500,300),labels=s1,title="Test Probabilities for Different
s")
    plot!(psamp2,labels=s2)
end
```

Simulated time series of observed positive tests for different s



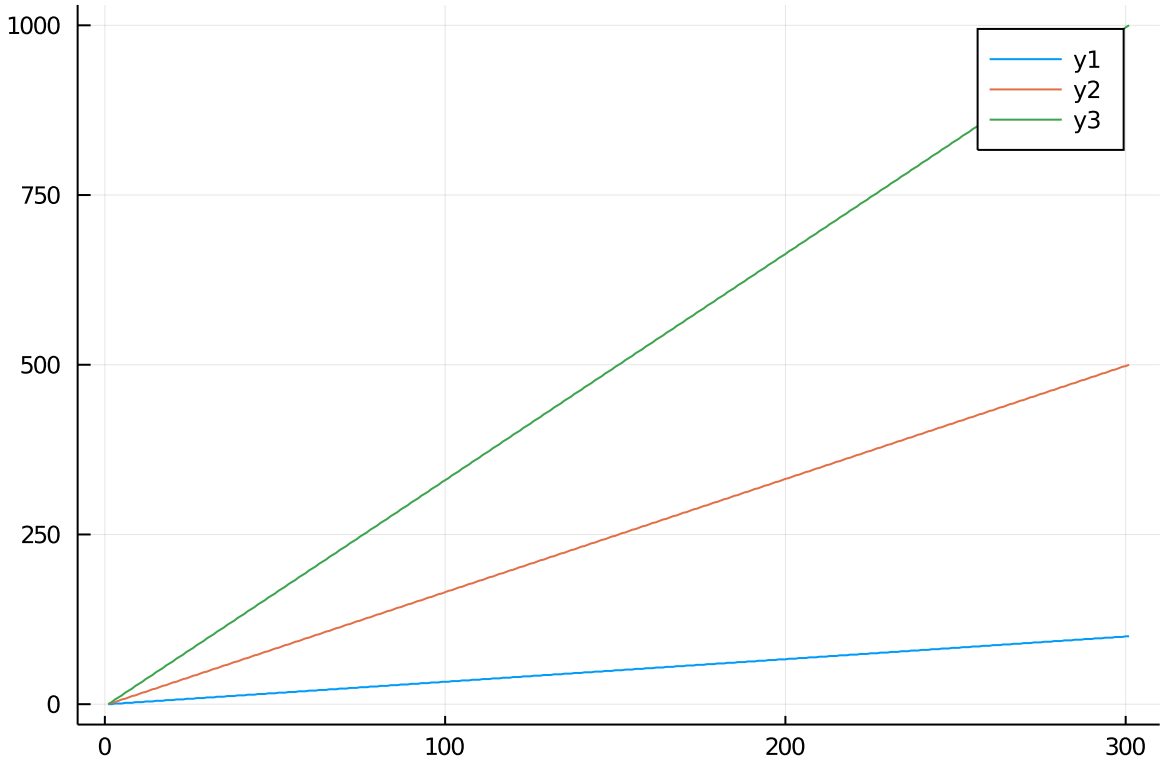

```
begin
    binoms1 = Binomial.(Ntest1,psamp1)
    binoms2 = Binomial.(Ntest1,psamp2)
    ts1 = rand.(binoms1)
    ts2 = rand.(binoms2)
    plot(ts1,size=(500,300),labels=s1)
    plot!(ts2,labels=s2)
```

```
end
```

TIME-VARYING N_{test}

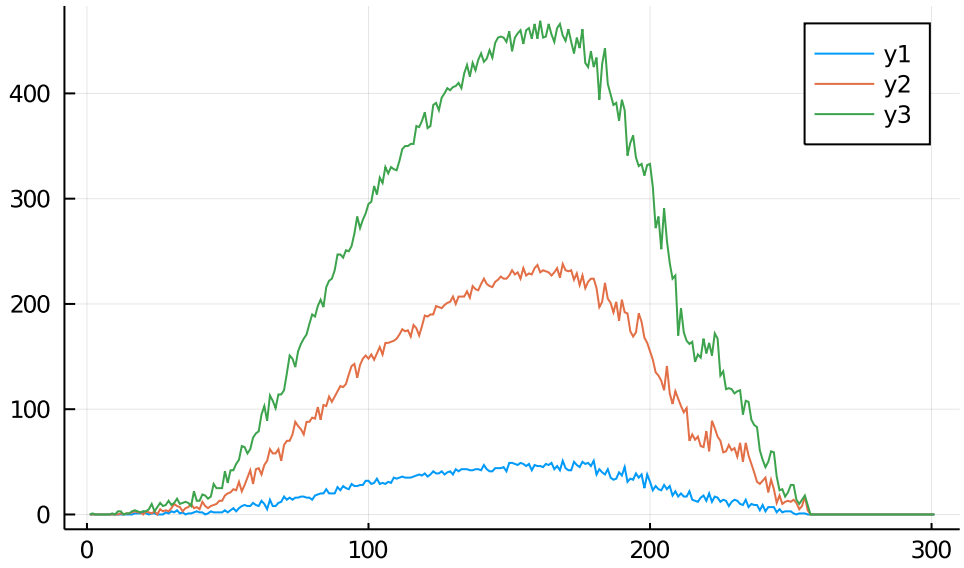
Linear increase in testing over time

We can impose time-dependent testing regimes to investigate its role in controlling observed growth rates



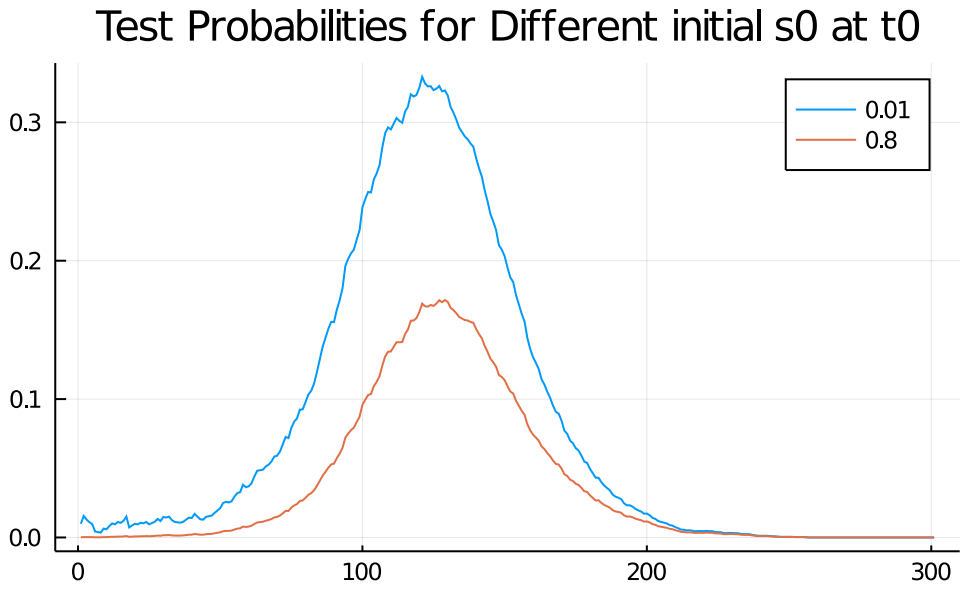

```
begin
    plot(Ntestlin1)
    plot!(Ntestlin2)
    plot!(Ntestlin3)
end
```

Plot observed positive tests for different Ntest time series




```
begin
    Ntestlin1 = convert.(Int64,round.(LinRange(0.0, 100.0, n),digits=0))
    Ntestlin2 = convert.(Int64,round.(LinRange(0.0, 500.0, n),digits=0))
    Ntestlin3 = convert.(Int64,round.(LinRange(0.0, 1000.0, n),digits=0))
    binomlin1 = Binomial.(Ntestlin1,psamp2)
    binomlin2 = Binomial.(Ntestlin2,psamp2)
    binomlin3 = Binomial.(Ntestlin3,psamp2)
    tlin1 = rand.(binomlin1)
    tlin2 = rand.(binomlin2)
    tlin3 = rand.(binomlin3)
    plin1 = plot(tlin1,size=(500,300))
    plin2 = plot!(tlin2)
    plin3 = plot!(tlin3)
end
```

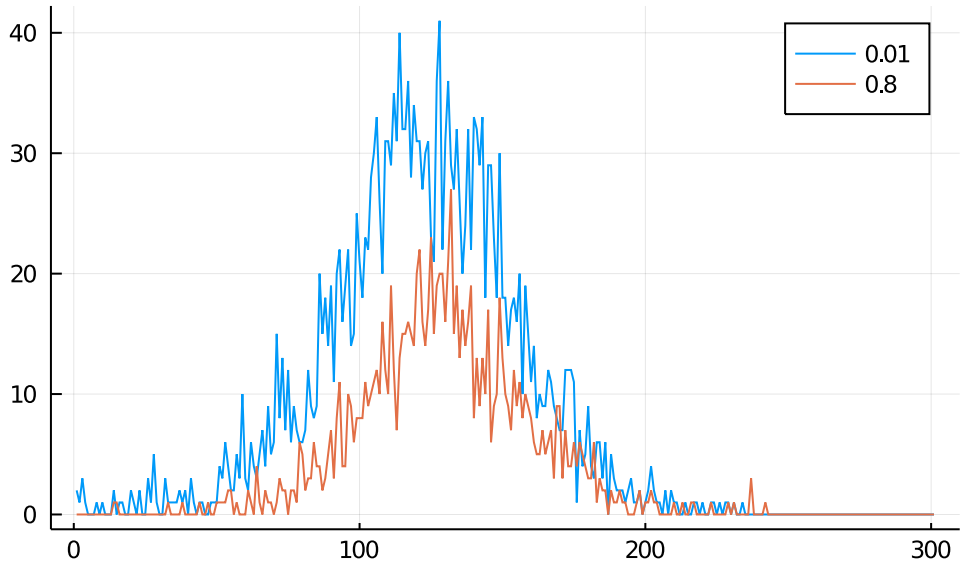
Linear increase in s over time




```
begin
    s01 = 0.01
    s02 = 0.8
```

```
    ss1 = LinRange(0.01, 0.8, n)
    ss2 = LinRange(0.8, 0.8, n)
    psamps1 = I./(I .+ ss1.*(Npop.-I))
    psamps2 = I./(I .+ ss2.*(Npop.-I))
    plot(psamps1,size=(500,300),labels=s01,title="Test Probabilities for Different initial s0 at t0")
    plot!(psamps2,labels=s02)
end
```

Plot simulated positive test time series for different s0




```
begin
    binomss1 = Binomial.(Ntest1,psamps1)
    binomss2 = Binomial.(Ntest1,psamps2)
    tss1 = rand.(binomss1)
    tss2 = rand.(binomss2)
    pss1 = plot(tss1,size=(500,300),label=s01)
    pss2 = plot!(tss2,label=s02)
end
```

Massachusetts data for example

	date	state	dataQualityGrade	death	deathConfirmed	deathIncrease	deat
4	"11/21/2020"	"MA"	"A+"	10488	10257	19	231
5	"11/20/2020"	"MA"	"A+"	10469	10238	34	231
6	"11/19/2020"	"MA"	"A+"	10435	10204	28	231

7	"11/18/2020"	"MA"	"A+"	10407	10177	47	230
8	"11/17/2020"	"MA"	"A+"	10360	10130	20	230
9	"11/16/2020"	"MA"	"A+"	10340	10110	11	230

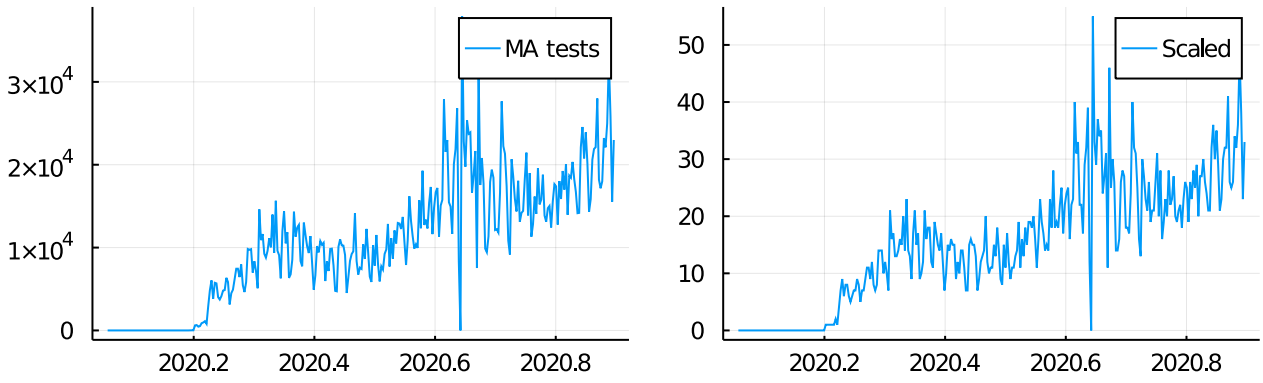
```
M = CSV.read("d:/google/working/covid/massachusetts-history.csv")
```

```
SentinelArrays.ChainedVector{String,Array{String,1}}: ["11/24/2020", "11/23/2020", "11/2
```

```
begin
    pos = M.positiveIncrease
    neg = M.negativeIncrease
    tests = pos + neg
    date = M.date
end
```

```
begin
    const DTM = Union{Date, DateTime} # 2000-01-01 to become 2000.0
    yfrac(dtm::DTM) = (dayofyear(dtm) - 1) / daysinyear(dtm)
    decimaldate(dtm::DTM) = year(dtm) + yfrac(dtm)

    time = zeros(0)
    for i=1:length(date)
        sp = split(date[i],"/")
        dd = Date(parse(Int,sp[3]),parse(Int,sp[1]),parse(Int,sp[2]))
        append!(time,decimaldate(dd))
    end
end
```




```
begin
    p9 = plot(time,tests)
    tests_N = convert.(Int64, round.((Npop/6.893E6)*tests, digits=0))
```

```
    p10 = plot(time,tests_N)
    plot(p9,p10,size=(650,200),labels=["MA tests" "Scaled"])
end
```