Exercise 1.10

Consider the data in the file `CRMdata.txt` which represents weekly claims (in million pesos) in the last 20 years for a certain insurance product, one week per row. It is only data for those weeks were claims were filed. Fit a collective risk model and estimate the expected value, variance, median, and $\mathrm{VaR}_{0.995}$ of the total claims per week.

```
# packages
using DataFrames , Distributions , Statistics , StatsBase , Plots ,
LaTeXStrings
```

Read and prepare the data

```
["5.411,5.364,5.664,7.04,14.799", "8.106,5.883,17.226,14.412,6.183,7.598,7.29,7.693,9.129
```

```
begin
    f = open("CRMdata.txt")
    rawdata = readlines(f)
    close(f)
    rawdata
end
```

```
nrows = 262
```

```
nrows = length(rawdata)
```

```
"5.411,5.364,5.664,7.04,14.799"
```

```
rawdata[1]
```

```
["5.411", "5.364", "5.664", "7.04", "14.799"]
```

```
split(rawdata[1], ",")
```

```
[5.411, 5.364, 5.664, 7.04, 14.799]
```

```
parse.(Float64, split(rawdata[1], ","))
```

```
[[5.411, 5.364, 5.664, 7.04, 14.799], [8.106, 5.883, 17.226, 14.412, 6.183, 7.598, 7.29, 7.
```

```
begin
    data = []
    for i ∈ 1:nrows
        push!(data, parse.(Float64, split(rawdata[i], ",")))
    end
    data
end
```

Frequency modeling

```
Nobs =
[5, 38, 4, 1, 5, 1, 20, 6, 1, 4, 1, 5, 1, 1, 2, 1, 2, 1, 1, 9,    more ,1, 1, 1, 1, 1, 1, 1, :
```

```
Nobs = length.(data) # observed frequency values
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 14, 17, 20, 25, 33, 38]
```

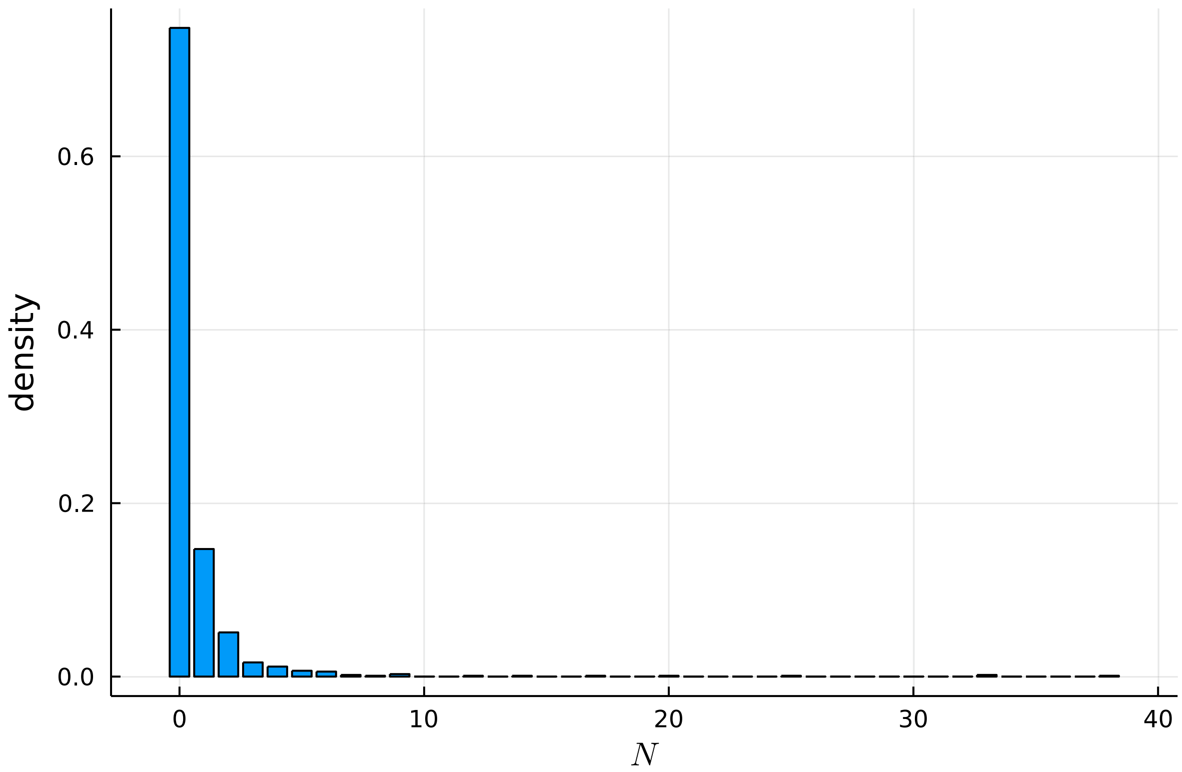
```
sort(unique(Nobs)) # unique observed values of frequency
```

	value	counts	proportion
1	0	778	0.748077
2	1	153	0.147115
3	2	53	0.0509615
4	3	17	0.0163462
5	4	12	0.0115385
6	5	7	0.00673077
7	6	6	0.00576923
8	7	2	0.00192308
9	8	1	0.000961538
10	9	3	0.00288462
	more		
39	38	1	0.000961538

```
begin # Frequency table
    Nvalues = collect(0:maximum(Nobs))
    Ncounts = zeros(Int, length(Nvalues))
    Ncounts[1] = 52*20 - nrows # number of cases when N = 0 (no claims)
    for n ∈ Nobs
        Ncounts[n+1] += 1
    end
    Nprop = Ncounts ./ sum(Ncounts)
    NfreqTable = DataFrame(value = Nvalues, counts = Ncounts, proportion = Nprop)
end
```

(1040, 1040, 1.0)

```
sum(Ncounts), 52*20, sum(NfreqTable.proportion) # just checking
```




```
bar(Nvalues, Nprop, xlabel = L"N", ylabel = "density", legend = false)
```

When the estimated probability $\mathbb{P}(N = 0)$ is significantly higher than for $N > 0$ it is better try fitting a model for $N|N > 0$. *To be continued...*

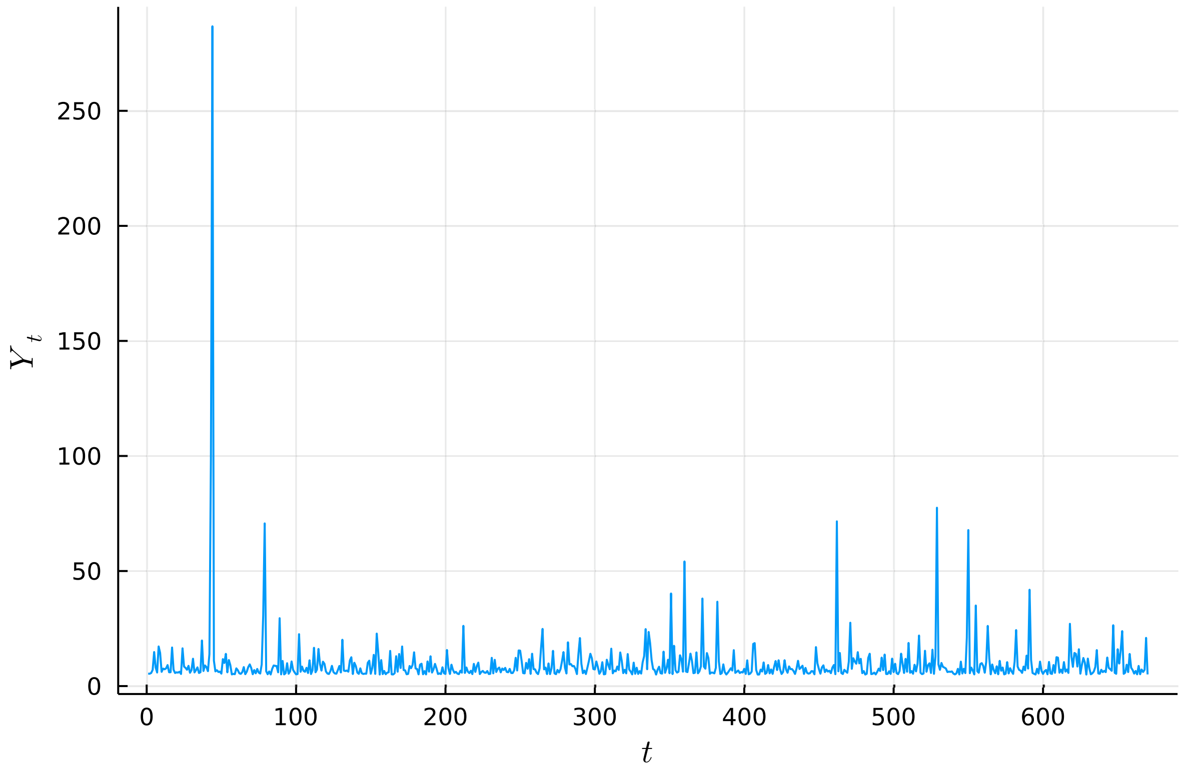
Severity modeling

```
[5.411, 5.364, 5.664, 7.04, 14.799, 8.106, 5.883, 17.226, 14.412, 6.183, 7.598, 7.29, 7.693
```

```julia
begin
    Yobs = zeros(0)
    for i ∈ 1:nrows
        append!(Yobs, data[i])
    end
    println("∑ N_i = ", sum(Nobs))
    Yobs # vector of claims in chronological order
end
```

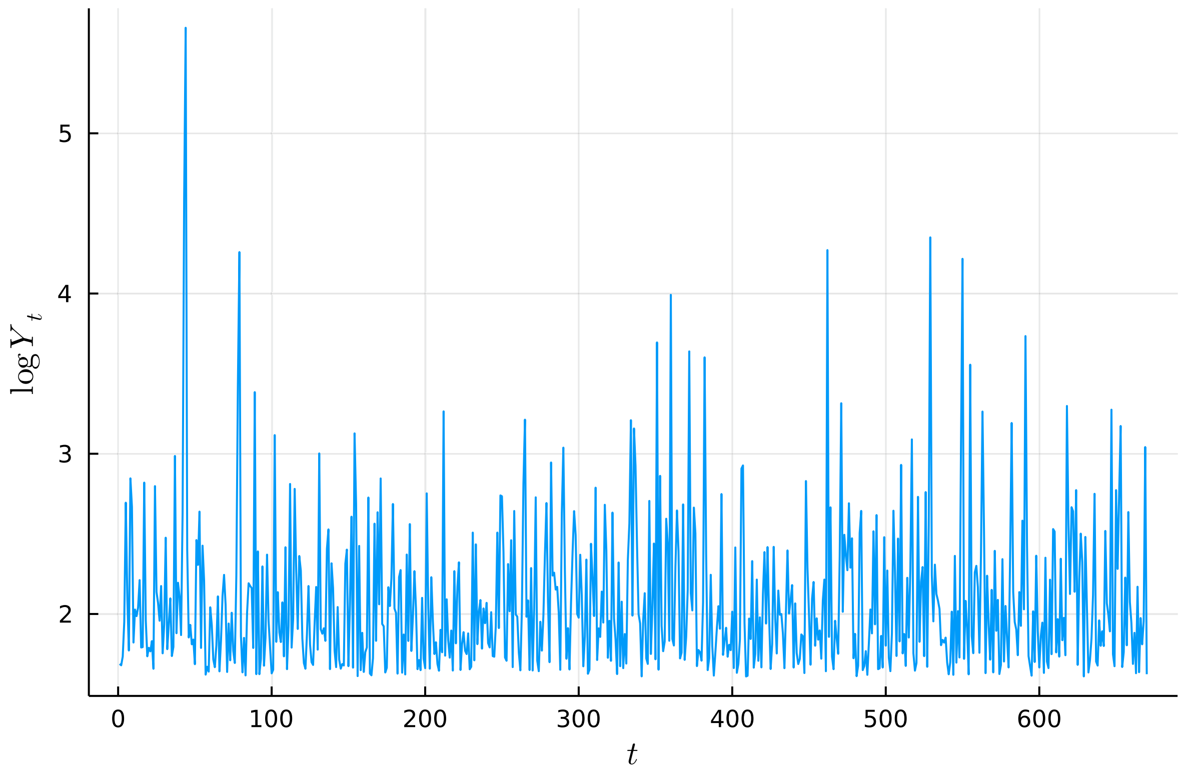
```
∑ N_i = 670
```

Check for serial independence of $Y_1, Y_2, \ldots$

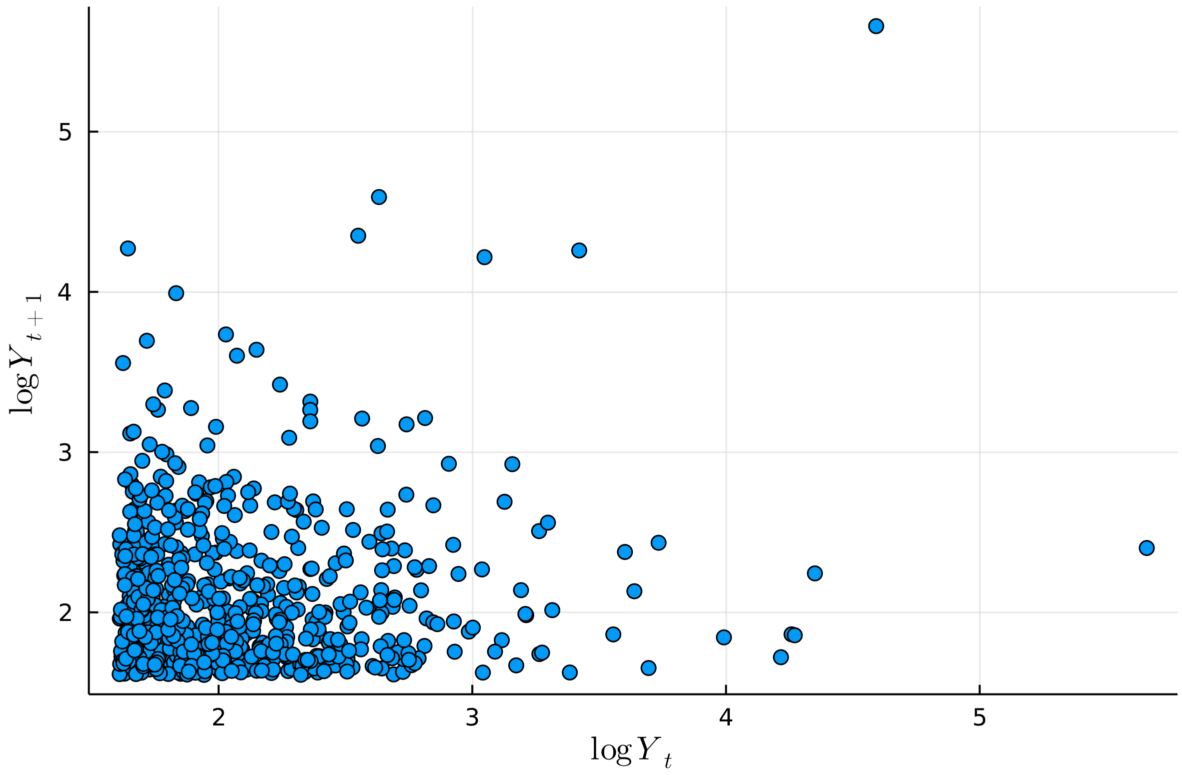



```julia
plot(Yobs, xlabel = L"t", ylabel = L"Y_t", legend = false)
```

For amounts of money a log-scale is preferred since vertical distances are proportional to % variations instead of absolute variations:




```julia
begin
    logYobs = log.(Yobs)
    plot(logYobs, xlabel = L"t", ylabel = L"\log\,Y_t", legend = false)
end
```



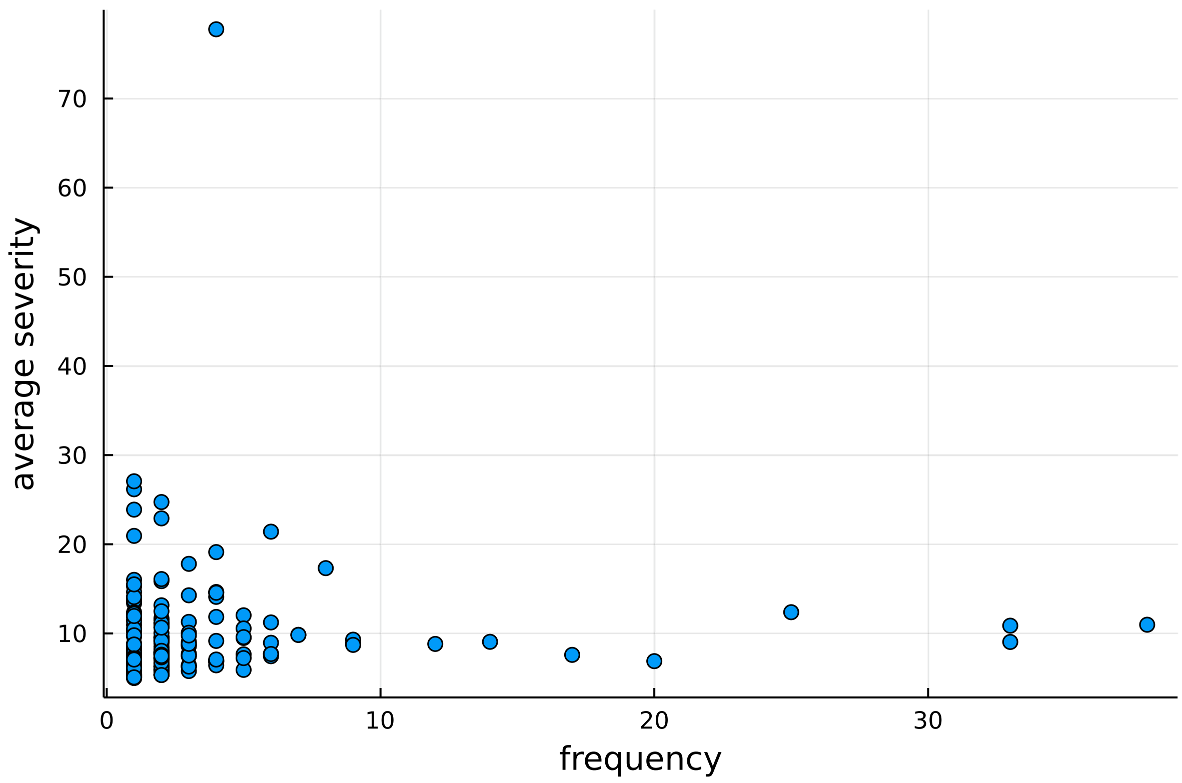

```
begin
    println("Spearman = ", corspearman(logYobs[1:(end-1)], logYobs[2:end])) # Spearman's correlation
    scatter(logYobs[1:(end-1)], logYobs[2:end], xlabel = L"\log\,Y_t", ylabel = L"\log\,Y_{t+1}", legend = false)
end
```

```
Spearman = -0.00027283981672650856
```

A Spearman correlation close to zero does not necessarily imply independence between claims $Y_1, Y_2, \ldots$ so statistical procedures must be applied to verify. *To be continued...*

Check for possible dependence between claim severity and frequency:

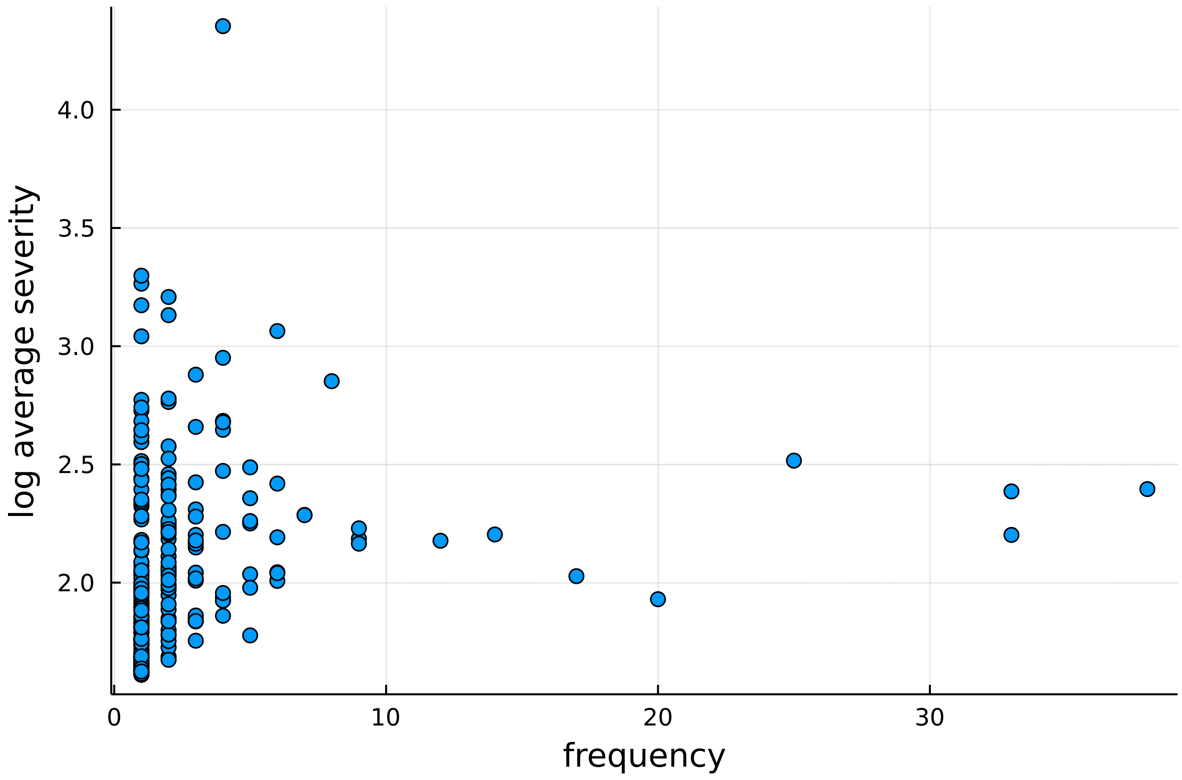
```
begin
    avgYobs = zeros(nrows)
    for i ∈ 1:nrows
        avgYobs[i] = mean(data[i])
    end
end
```




```
begin
    println("Spearman = ", corspearman(Nobs, avgYobs))
    scatter(Nobs, avgYobs, legend = false, xlabel = "frequency", ylabel = "average severity")
end
```

```
Spearman = 0.44433139528796955
```

The same as before but in log-scale:




```
begin
    println("Spearman = ", corspearman(Nobs, log.(avgYobs)))
    scatter(Nobs, log.(avgYobs), legend = false, xlabel = "frequency", ylabel = "log
    average severity")
end
```

```
Spearman = 0.44433139528796955
```

Spearman's correlation clearly different from zero implies there is NO independence between frequency and severity, so in this case it is necessary to find a model for the severity conditional on the frequency. *To be continued...*