Laboratorio di fisica computazionale

Homework 6

Author: Ian Gremese

```julia
begin
    #using Pkg;Pkg.update();Pkg.add("SpecialFunctions")
    using Random ✓
    using Formatting ✓
    using Printf ✓
    using Plots ✓
    import StatsBase ✓  as SB
    import LinearAlgebra ✓  as LA
    theme(:ggplot2)
    using PlutoUI ✓
    import SpecialFunctions ✓  as SF
    # using LaTeXStrings
end
```

The next function generates a random variable distributed uniformly on the interval $[-1, +1)$.

```
unipm1 (generic function with 1 method)
```

```julia
function unipm1()
    return 2*rand(Float64)-1
end
```

The next function generates a random variable distributed according to $p(x) = \exp(-x)$ on the interval $[-1, +1)$.

```
e⁻rnd (generic function with 1 method)
```

```julia
function e⁻rnd()
    return 0. - log1p( - rand(Float64) )
end
```

The next function generates a random variable distributed according to $p(x) = \frac{a}{\pi}\frac{1}{x^2+a^2}$.

```
Lorrnd (generic function with 1 method)
```

```julia
function Lorrnd() #(a::Float64)
    return 1. * tan(pi*(rand(Float64) - 1/2)) #1. = a
end
```

CLT

This function produces a K-uple of averages x of random variables distributed according to the function f.

Arguments:

- N: number of points for the average
- K: number of averages to be generated
- f: function which generates the N random variables r of which the average is calculated. If default is left, it generates a variable distributed accordingly to the uniform distribution on [-1,1).
- μ: analytical average of the distribution of the random variable r, produced by f(rand())
- σ: analytical variance of the distribution of the random variable r, produced by f(rand())

-extras: boolean variable that, when "true", triggers the computation of the ratio $\langle z^4\rangle/3/\langle z^2\rangle$, which is the returned as third output

Returns:

- x: list of K averages
- σ: estimate of the standard deviation of the K averages
- ratio $\langle z^4\rangle/3/\langle z^2\rangle$: ratio that should equal 1 for z according to N(0,1)

```
"""
This function produces a K-uple of averages x of random variables distributed
according to the function f.

Arguments:

- N: number of points for the average

- K: number of averages to be generated

- f: function which generates the N random variables r of which the average is
calculated. If default is left, it generates a variable distributed accordingly to
the uniform distribution on [-1,1).

- μ: analytical average of the distribution of the random variable r, produced by
f(rand())

- σ: analytical variance of the distribution of the random variable r, produced by
f(rand())

-extras: boolean variable that, when "true", triggers the computation of the ratio
⟨z⁴⟩/3/⟨z²⟩, which is the returned as third output

Returns:

```

```
- x: list of K averages

- σ: estimate of the standard deviation of the K averages

- ratio ⟨z⁴⟩/3/⟨z²⟩: ratio that should equal 1 for z according to N(0,1)
"""
function
CLT(N::Int64,K::Int64;f=unipm1::Function,μ₀=0.::Float64,σ₀=1.::Float64,extras=false:
:Bool)

    x = zeros(Float64,K)
    μ = 0.::Float64
    σ = 0.::Float64
    z̄⁴ = 0.::Float64
    z̄² = 0.::Float64
    for j = 1:1:K
        xhere = 0.
        for i = 1:1:N
            r = f()
            xhere += r
        end
        xhere = xhere/N
        x[j] = xhere
        μ = μ + xhere
        σ = σ + xhere^2

        if (extras == true)
            z = (xhere-μ₀)/σ₀
            z̄⁴ += z^4
            z̄² += z^2
        end

    end

    μ = μ/K
    σ = (σ/K-μ^2)*sqrt(K/(K-1))
    σ = sqrt(σ)

    if (extras == true)
        z̄⁴ = z̄⁴/K
        z̄² = z̄²/K
        ratio = z̄⁴/3/(z̄²)^2
        return x, σ, ratio
    end

    if (extras == false)
        return x, σ
    end

    # plot(xₙ,seriestype = :histogram)

end
```

Averages of uniform random numbers in $[-1, +1)$

```julia
begin
    N₁ = 500::Int64
    K₁ = 100::Int64

    x₁, σ₁, one₁ = CLT(N₁,K₁;μ₀ = 0.,σ₀ = 1/sqrt(3*N₁),extras = true)
    x̄₁ = sum(x₁)/K₁
end;
```

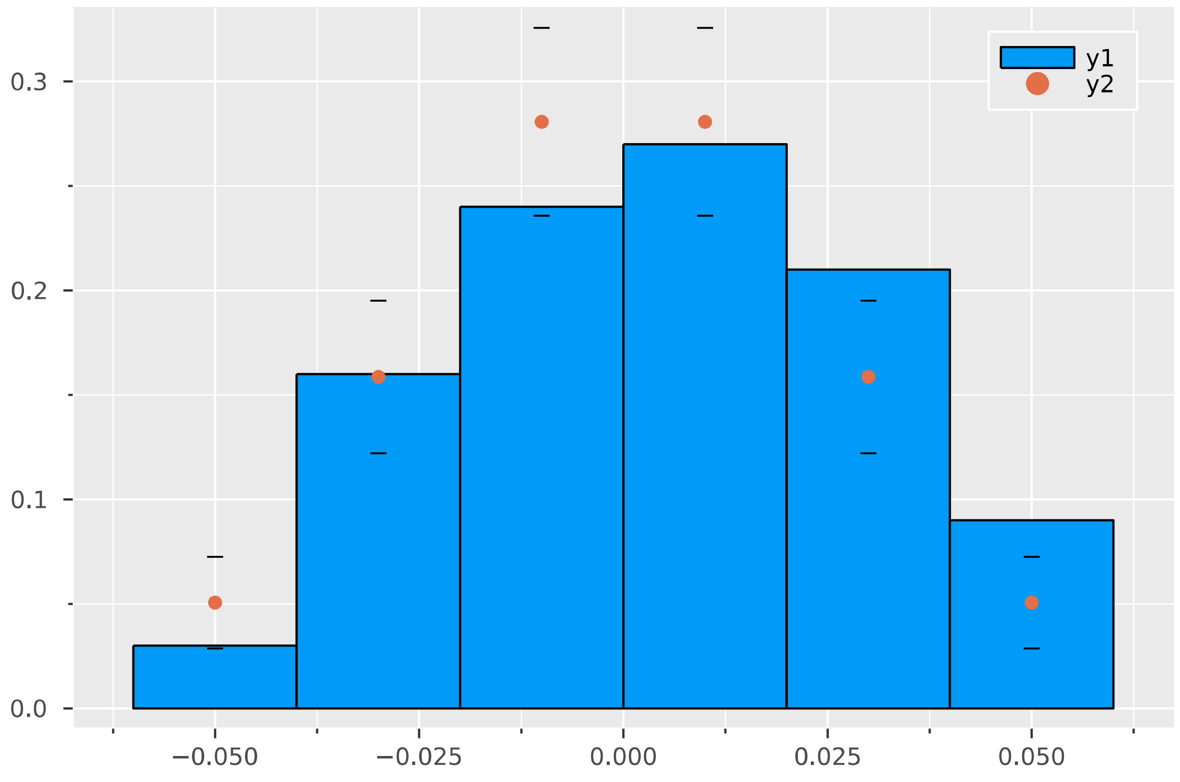
```julia
begin
    x1fmt = @sprintf "%.3f" x̄₁
    σ1fmt = @sprintf "%.3f" σ₁
    one1fmt = @sprintf "%.3f" one₁

    #theoretical value
    σ1th = @sprintf "%.3f" 1/sqrt(3*N₁)
end;
```

The values for K averages over N uniform random numbers r in $[-1,+1)$ we obtained, written as `exp. value -> theor. value` are:

- average: 0.005 -> 0.0;
- std. dev.: 0.025 -> 0.026;
- $\langle z^4\rangle/3\langle z^2\rangle^2$: 0.755 -> 1;

the theoretical values having been computed respectively as the expectation value of the rs and the std. dev. of the original rs, $\sqrt{1/3}$, divided by $\sqrt{N}$, as required by the formula of the propagation of the variance.



```julia
begin
    # Automated histogram generation, from which we get the bins
    local h = SB.fit(SB.Histogram, x₁)#, nbins=10)
    local plotrange = h.edges[1]

    # We use the bins to construct the theoretical normalized histogram
    # using the gaussian error function
        local σ = 1/sqrt(3*N₁)
        local Δₚₗₒₜ = plotrange[2]-plotrange[1]

        # Broadcast the function ''1/2 * (erf((x+Δₚₗₒₜ)/(sqrt(2)*σ))-
        erf(x/(sqrt(2)*σ)))''
        # onto the array plotrange[1:end-1] to compute the normalized histogram
        local plotfunc = broadcast(x -> 1/2 * (SF.erf((x+Δₚₗₒₜ)/(sqrt(2)*σ))-
        SF.erf(x/(sqrt(2)*σ))), plotrange[1:end-1])

        local plotcentres = [(plotrange[i]+plotrange[i-1])/2 for i in
        2:1:length(plotrange)]

    # Final plot
    h = LA.normalize(h; mode=:probability)
    plot(h)
    plot!(plotcentres, plotfunc, yerr = sqrt.(plotfunc .* (1 .- plotfunc)/K₁),
    seriestype = :scatter)
end
```

One can see that the experimental values and the theoretical ones seem to be in good agreement with each other. The histogram and the function plot, too, agree within 1-2 standard deviation with each other in most runs, providing evidence for the validity of the Central Limit Theorem.

Averages of random numbers with $p(x) = e^{-x}$

```julia
md"""#### Averages of random numbers with ``p(x) = e^{-x}``"""
```

```
begin
    N₂ = 500::Int64
    K₂ = 100::Int64

    x₂, σ₂, one₂ = CLT(N₂,K₂;f=e⁻rnd,μ₀ = 1.,σ₀ = 1/sqrt(N₂), extras = true)
    x̄₂ = sum(x₂)/K₂
end;
```

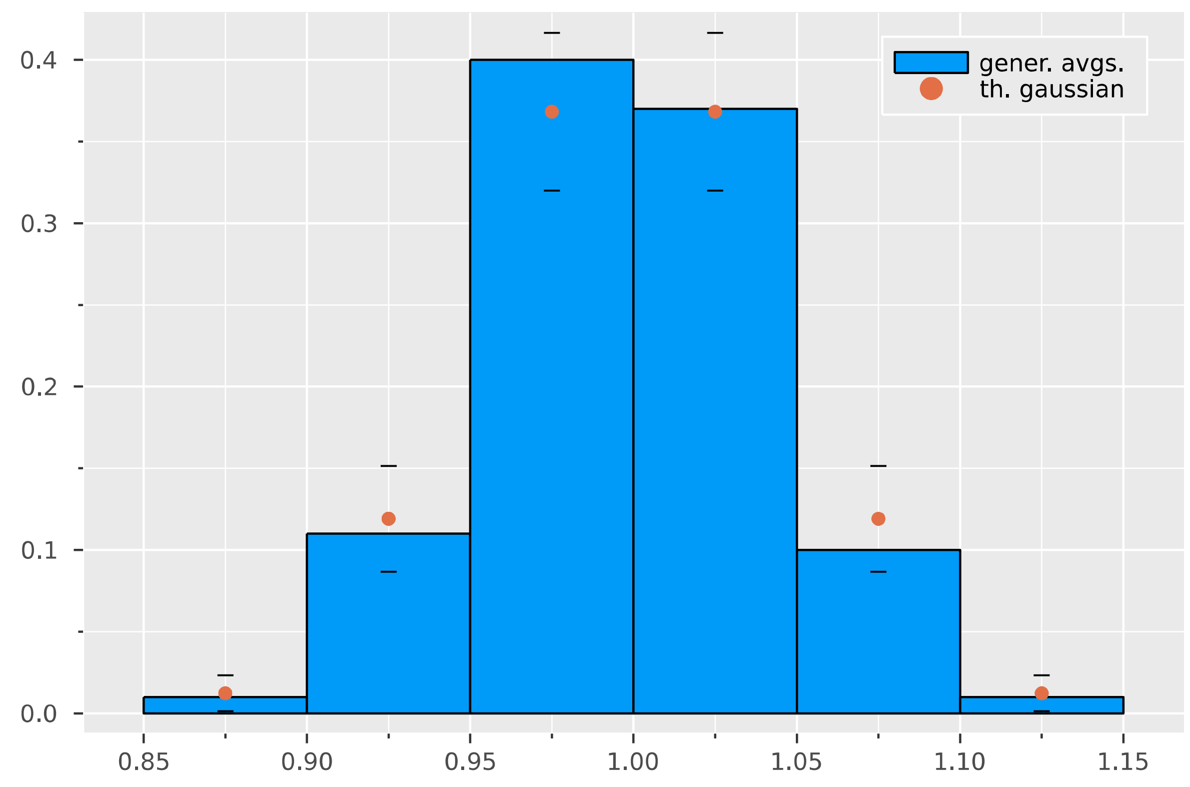
```
begin
    # Exp. values that are printed in the next cell
    x2fmt = @sprintf "%.3f" x̄₂
    σ2fmt = @sprintf "%.3f" σ₂
    one2fmt = @sprintf "%.3f" one₂

    # Theoretical value for the std. dev.
    σ2th = @sprintf "%.3f" 1. /sqrt(N₂)
end;
```

The values for K averages over N random numbers r distributed accordingly to $p(x) = e^{-x}$ we obtained, written as `exp. value -> theor. value` are:

- average: 0.999 -> 1.0;
- std. dev.: 0.043 -> 0.045;
- $\langle z^4 \rangle / 3 \langle z^2 \rangle^2$: 1.086 -> 1;

the theoretical values having been computed respectively as the expectation value of the rs and the std. dev. of the original rs, 1, divided by $\sqrt{N}$, as required by the formula of the propagation of the variance.



```
begin
    # Automated histogram generation, from which we get the bins
    local h = SB.fit(SB.Histogram, x₂)#, nbins=10)
    local plotrange = h.edges[1]

    # We use the bins to construct the theoretical normalized histogram
    # using the gaussian error function
        local σ = 1/sqrt(N₂)
        local Δₚₗₒₜ = plotrange[2]-plotrange[1]

        # Broadcast the function ''1/2 * (erf((x+Δₚₗₒₜ)/(sqrt(2)*σ))-
        erf(x/(sqrt(2)*σ)))''
        # onto the array plotrange[1:end-1] to compute the normalized histogram
        local plotfunc = broadcast(x -> 1/2 * (SF.erf((x+Δₚₗₒₜ-1.)/(sqrt(2)*σ))-
        SF.erf((x-1.)/(sqrt(2)*σ))), plotrange[1:end-1])

        local plotcentres = [(plotrange[i]+plotrange[i-1])/2 for i in
        2:1:length(plotrange)]

    # Final plot
    h = LA.normalize(h; mode=:probability)
    plot(h, label = :"gener. avgs.")
    plot!(plotcentres, plotfunc, yerr = sqrt.(plotfunc .* (1 .- plotfunc)/K₁),
    seriestype = :scatter,label = "th. gaussian")
end
```

The histogram and the theoretical prediction for the counts for the new gaussian average agree with each other within 1-2 standard deviations on most runs, except for bins with very low counts, providing further evidence for the validity of the Central Limit Theorem.

Averages of random numbers with $p(x) = 1/\pi/(x^2+1)$

```
begin
    N₃ = 500::Int64
    K₃ = 100::Int64

    x₃, σ₃ = CLT(N₃,K₃;f = Lorrnd)
    x̄₃ = sum(x₃)/K₃
end;
```

```
begin
    # Exp. values that are printed in the next cell
    x3fmt = @sprintf "%.3f" x̄₃
    σ3fmt = @sprintf "%.3f" σ₃
end;
```

Using the characteristic function of the lorentzian distribution one can show that, if x is distributed accordingly to the lorentzian, which has the characteristic function

$$\Phi_X(t) = \exp\left(-a|t|\right) = E\left[e^{itx}\right] = \int_{-\infty}^{+\infty} dx\, \frac{a}{\pi}\frac{1}{x^2+a^2}e^{itx} \ ,$$

the average $\bar{x}$ on N independent xs will be distributed according to a distribution function with characteristic function

$$\Phi_{\bar{X}}(t) = E\left[e^{it\bar{x}}\right] = E\left[e^{it\sum_{j=1}^{N} x/N}\right] = \left(E\left[e^{itx/N}\right]\right)^N =$$

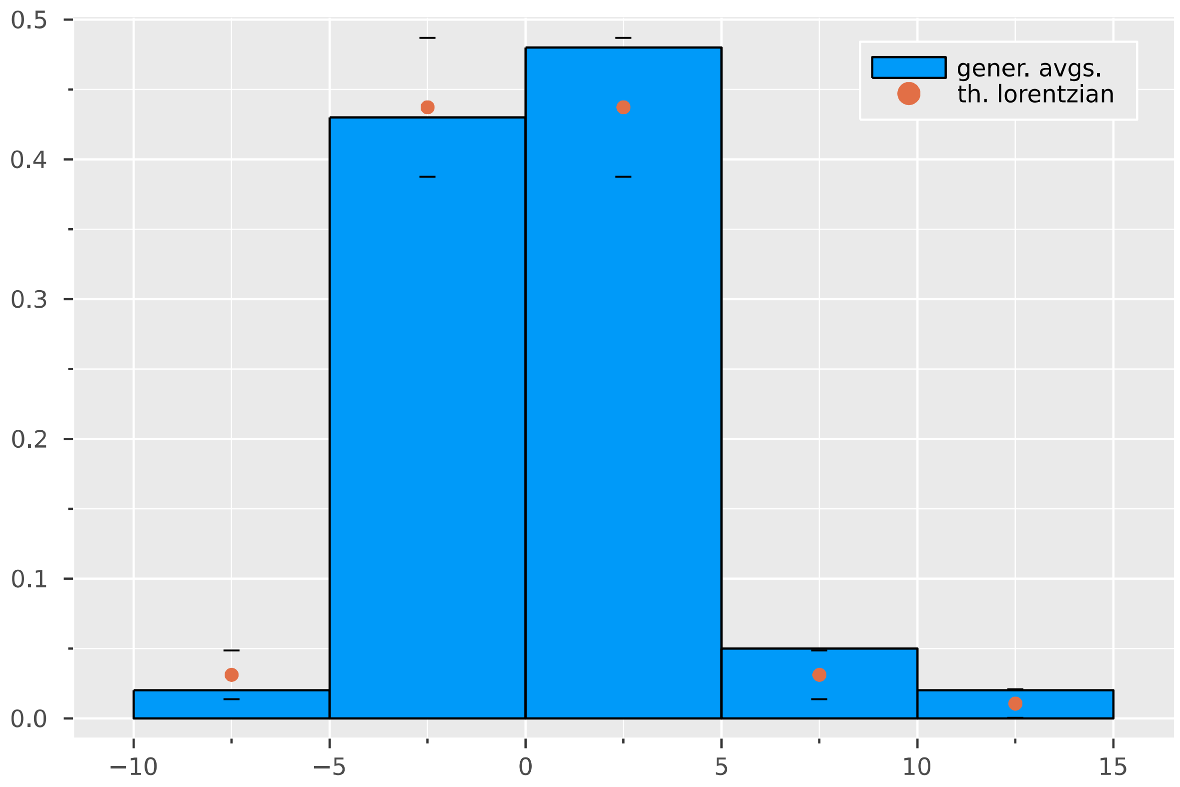
$$\left(\int_{-\infty}^{+\infty} dx\, \frac{a/N}{\pi}\frac{1}{x^2+(a/N)^2}e^{itx}\right)^N = \left(\exp\left(-\frac{a}{N}|t|\right)\right)^N = \Phi_X(t) \ ,$$

which means the average itself will be a lorentzian variable with the same parametre a as the xs from which it was computed.

The values for K averages over N random numbers r distributed accordingly to the lorentzian $p(x) = \frac{1}{\pi}\frac{1}{x^2+1}$ we obtained, written as `exp. value -> theor. value` are:

- average: 0.526 -> 0.0;
- std. dev.: 2.634 -> Inf;

the theoretical values having been computed respectively as the expectation value of the rs and the std. dev. of the original rs, 1, divided by $\sqrt{N}$, as required by the formula of the propagation of the variance.



```
begin
    # Automated histogram generation, from which we get the bins
    local h = SB.fit(SB.Histogram, x₃)#, nbins=10)
    local plotrange = h.edges[1]

    # We use the bins to construct the theoretical normalized histogram
    # using the gaussian error function
        local σ = 1/sqrt(N₃)
        local Δₚₗₒₜ = plotrange[2]-plotrange[1]

        # Broadcast the function ''1/2 * (erf((x+Δₚₗₒₜ)/(sqrt(2)*σ))-
        erf(x/(sqrt(2)*σ)))''
        # onto the array plotrange[1:end-1] to compute the normalized histogram
        local plotfunc = broadcast(x -> 1/pi * ( atan((x+Δₚₗₒₜ)/1.) - atan(x/1.) ),
        plotrange[1:end-1])

        local plotcentres = [(plotrange[i]+plotrange[i-1])/2 for i in
        2:1:length(plotrange)]

    # Final plot
    h = LA.normalize(h; mode=:probability)
    plot(h, label = :"gener. avgs.")
    plot!(plotcentres, plotfunc, yerr = sqrt.(plotfunc .* (1 .- plotfunc)/K₁),
    seriestype = :scatter,label = "th. lorentzian")
end
```

The histogram and the theoretical prediction for the counts for the lorentzian average agree within 1-2 standard deviations with each other on most of the runs, except for bins with very low counts. Comparing the theoretical values to the experimental ones, though, one gets disappointing results because of how likely results far from the average are, which is what causes the value of the variance to be high. One might expect to get better estimates of the average by increasing the number of averages that one generates, but that doesn't seem to happen.