Table of Contents

In [34]:
```
using LinearAlgebra, Random, Plots
```

In [35]:
```
default(fmt=:png)
```

In [36]:
```
function rand_graph(n)
    M = triu(rand(0:1,n,n),1)
    M = M + M'
end
```

Out[36]:
```
rand_graph (generic function with 1 method)
```

In [37]:
```
Random.seed!(1)
rand_graph(4)
```

Out[37]:
```
4×4 Matrix{Int64}:
 0  1  1  0
 1  0  0  0
 1  0  0  1
 0  0  1  0
```

Let's look at the adjacency eigenvalues of a random graph on 20 vertices.

In [38]:
```
n = 20
M = rand_graph(n)
e = eigvals(M)
```

Out[38]:
```
20-element Vector{Float64}:
 -4.4167213562514895
 -3.6154886618075093
 -3.143202870373745
 -2.5825051468758127
 -2.0955481620431256
 -1.8109701640759586
 -1.435348272404787
 -1.3236040830426876
 -0.8306963356626946
 -0.6814729400829646
 -0.5416890575034554
  0.06368665785353296
  0.5617713297606726
  0.6383584605767725
  1.3147452784500198
  1.9270987242512794
  2.078487712487037
  2.494786568261749
  3.4227317405805486
  9.975580577902623
```

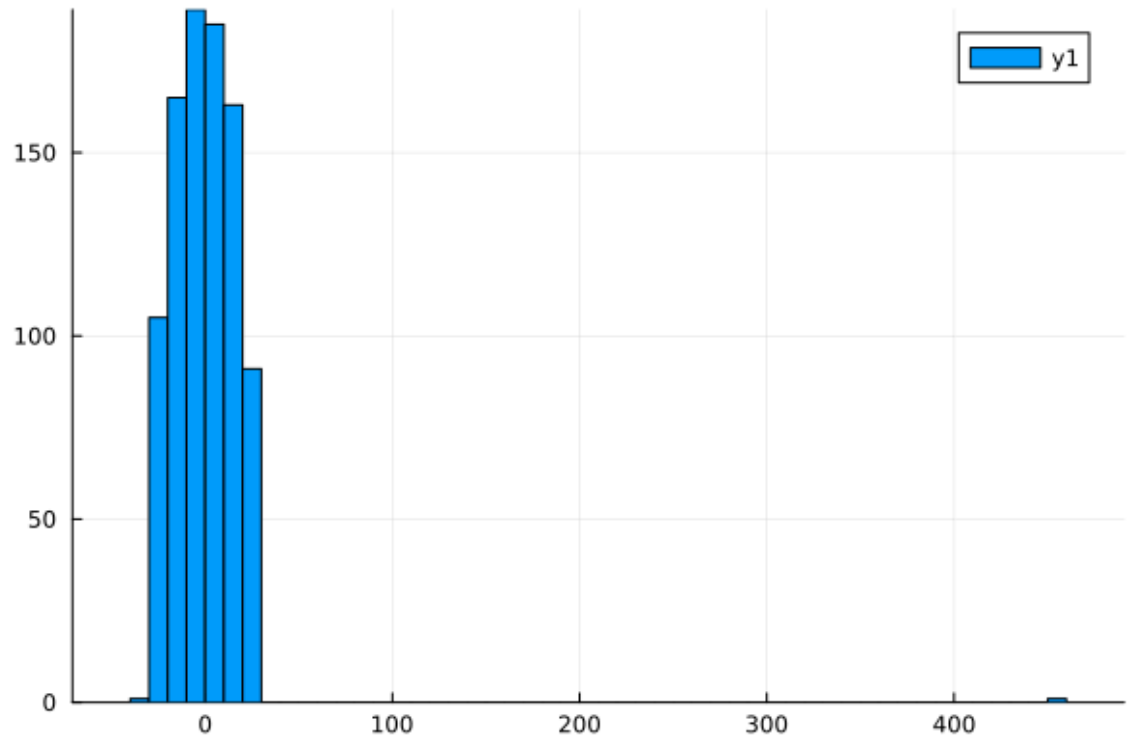
Let's look at the adjacency eigenvalues of a random graph on 1000 vertices.

In [39]:
```
n = 900
M = rand_graph(n)
e = eigvals(M)
```

Out[39]:
```
900-element Vector{Float64}:
 -30.348898526858854
 -29.87534246979177
 -29.77505500818062
 -29.58746083629446
 -29.332861831140296
 -29.200806380944805
 -28.8490855527281
 -28.712451101408597
 -28.605598164646217
 -28.380886734130687
 -28.272015824492026
 -28.19937749761563
 -28.053337161846194
   ⋮
  27.175526497205777
  27.411703988767094
  27.570636481766563
  27.580457531863637
  27.859693560442313
  27.943379266768794
  28.090935794511715
  28.35028074026666
  28.46185702662848
  28.631398005876267
  28.992915309826046
 450.2691567270127
```

In [40]:
```
histogram(e,bins=50)
```

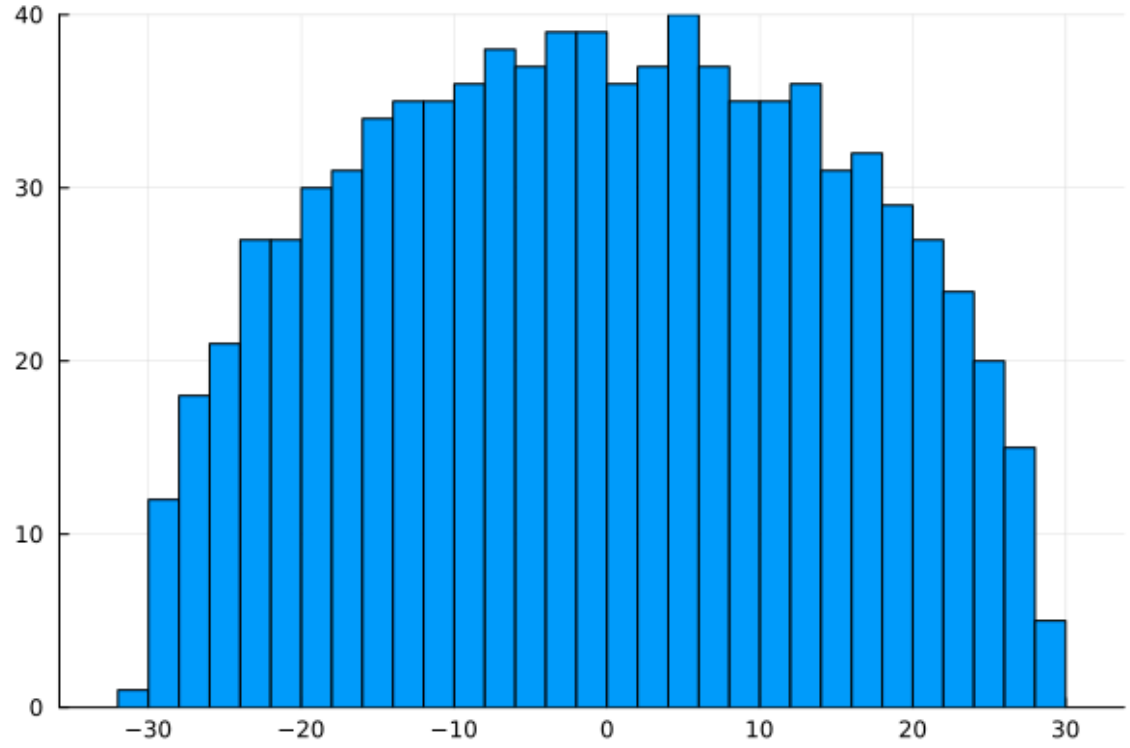
Out[40]:



We see that there is one big eigenvalue, and the the others are concentrated around -30 to 30. Let's just focus on those. And, note that 30 is the square root of 900.

In [41]:
```
histogram(e[1:(n-1)], bins=50, legend=false)
```

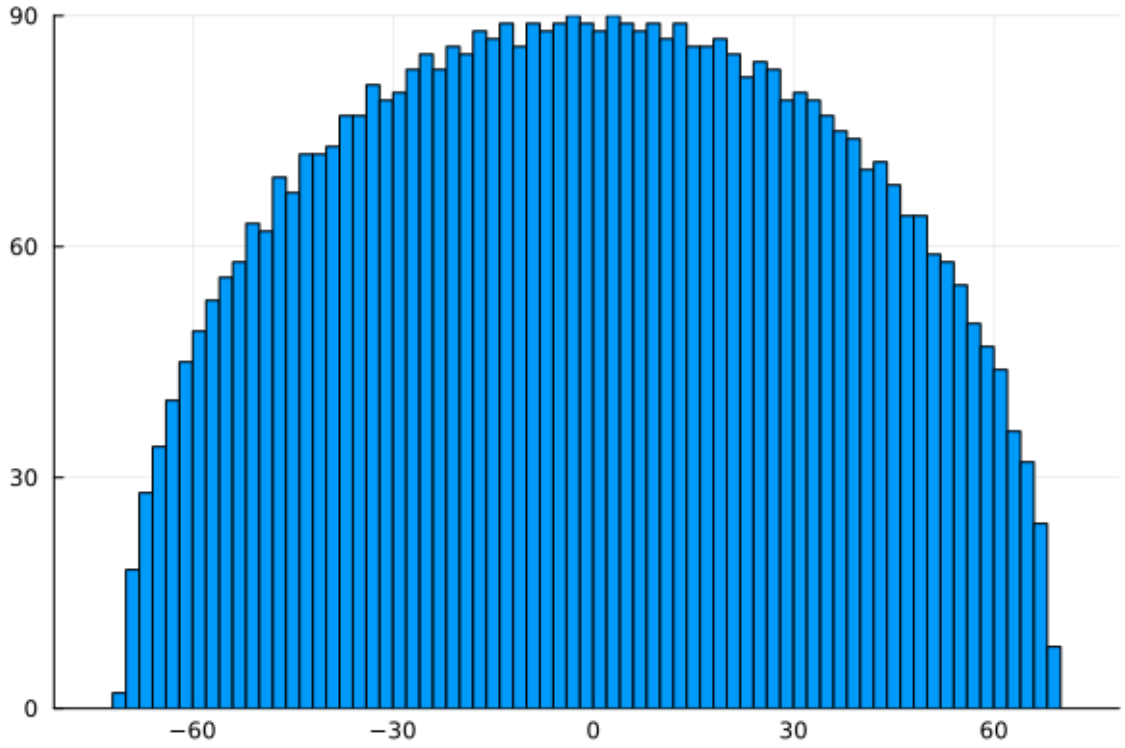
Out[41]:



It turns out that these have a very disinct distribution. It is called the semicircle law. To see it better, let's try a larger matrix.

In [42]:
```
n = 4900
M = rand_graph(n)
e = eigvals(M)
histogram(e[1:(n-1)], bins=100, legend=false)
```

Out[42]:



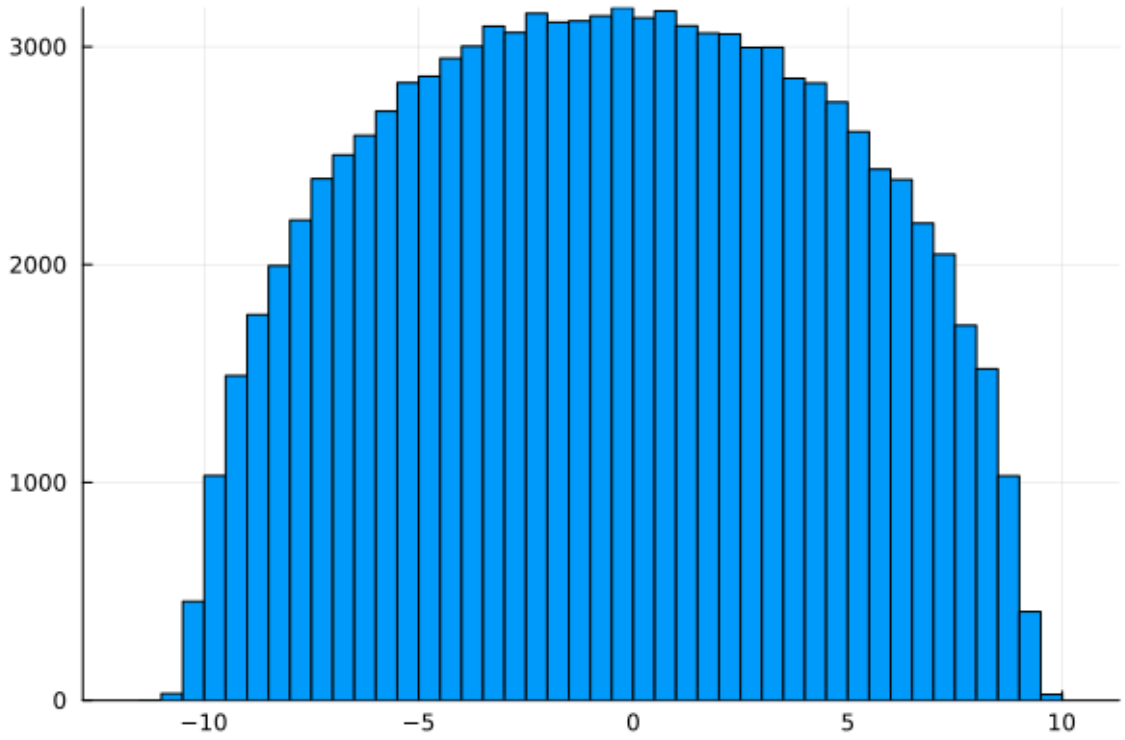
In [44]:
```
e[1], e[4900-1], e[end]
```

Out[44]:
```
(-70.53122628439453, 69.26387884870459, 2450.5126319439623)
```

Or, we could generate many matrices, and combine their eigenvalues.

In [45]:
```
e = []
n = 100
for i in 1:1000
    M = rand_graph(n)
    append!(e,eigvals(M)[1:(n-1)])
end
histogram(e, bins=40, legend=false)
```

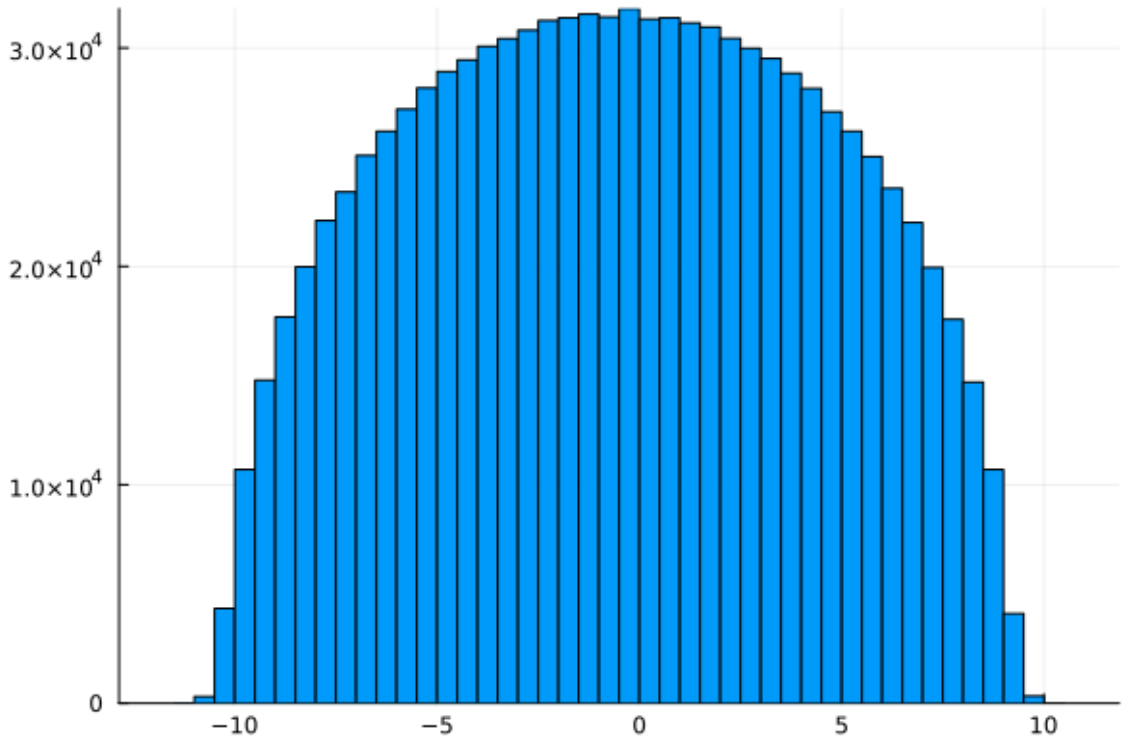
Out[45]:



In [13]:

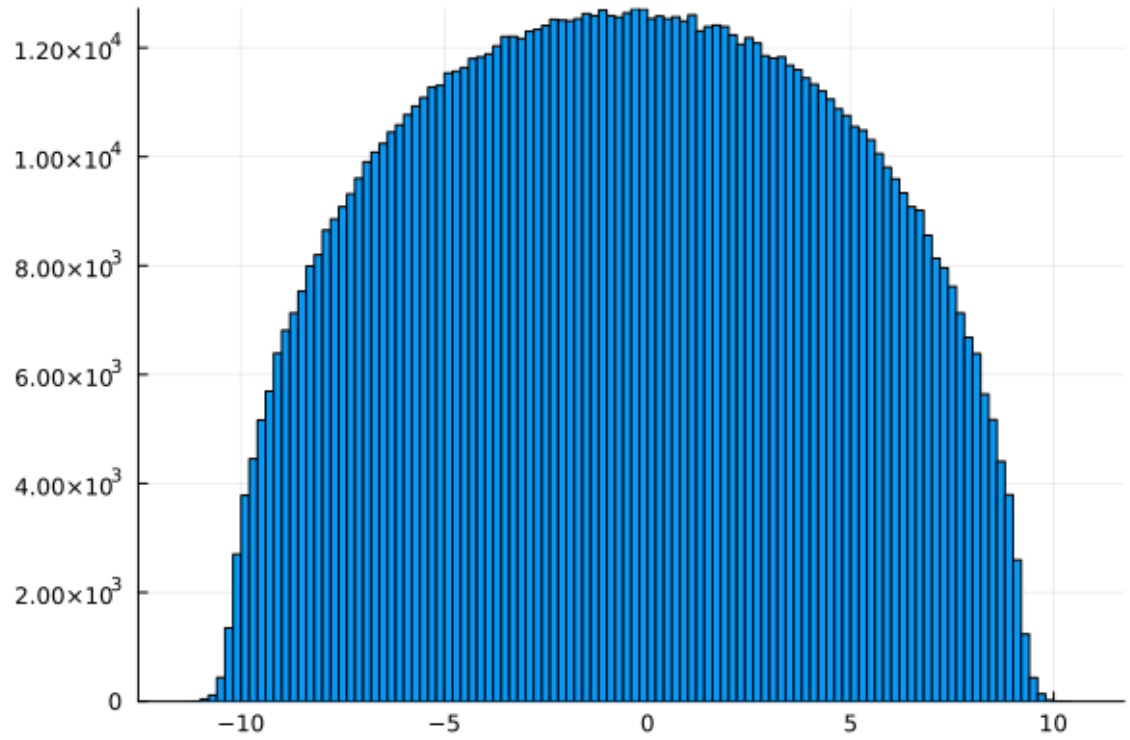
```
e = []
n = 100
for i in 1:10000
    M = rand_graph(n)
    append!(e,eigvals(M)[1:(n-1)])
end
histogram(e, bins=50, legend=false)
```

Out[13]:



In [46]:
```
es = []
n = 100
for i in 1:10_000
    M = rand_graph(n)
    append!(es,eigvals(M)[1:(n-1)])
end
h = histogram(es, legend=false)
```

Out[46]:

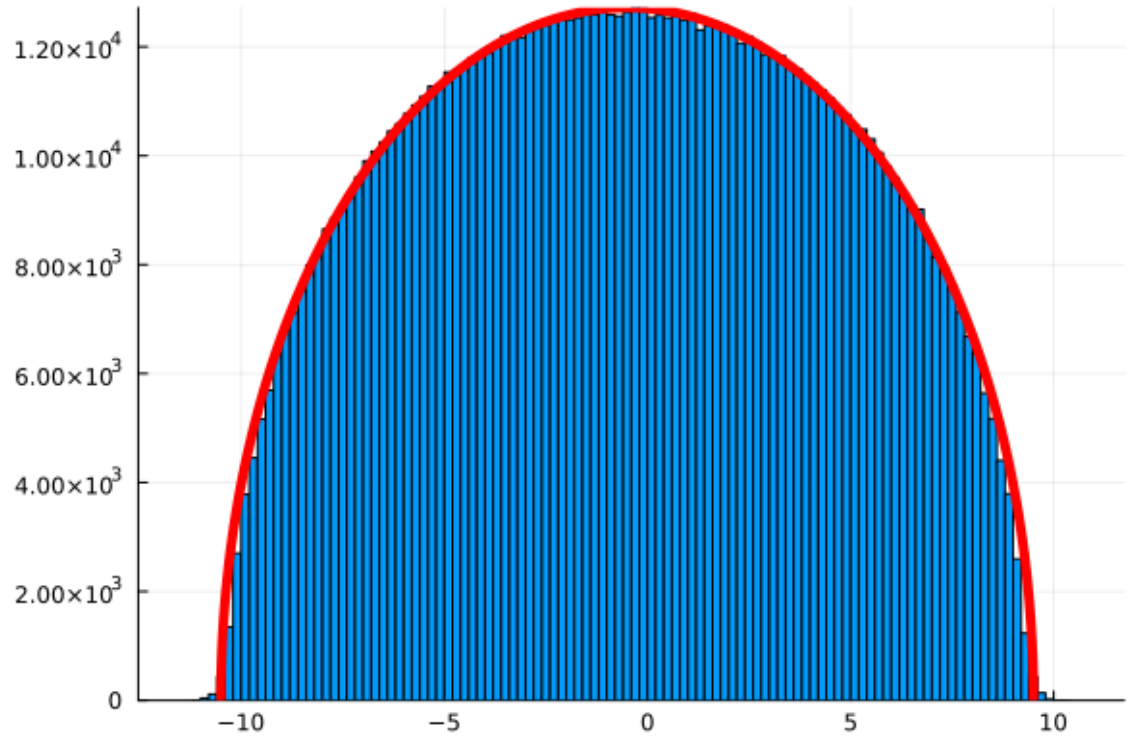


In [47]:
```
ydat = h[1][1][:y]
ymax = maximum(ydat[.!(isnan.(ydat))])
```

Out[47]:
```
12717.0
```

In [48]:
```
thetas = 0:.01:pi
xs = cos.(thetas)*10 .- 0.5
ys = sin.(thetas)*ymax
plot!(h, xs, ys, color="red", linewidth=5)
```

Out[48]:



Not only does the distribution have a very nice shape, but the minimum and maximum eigenvalues are almost never outside the edges of it.

In [49]:
```
minimum(es), maximum(es)
```

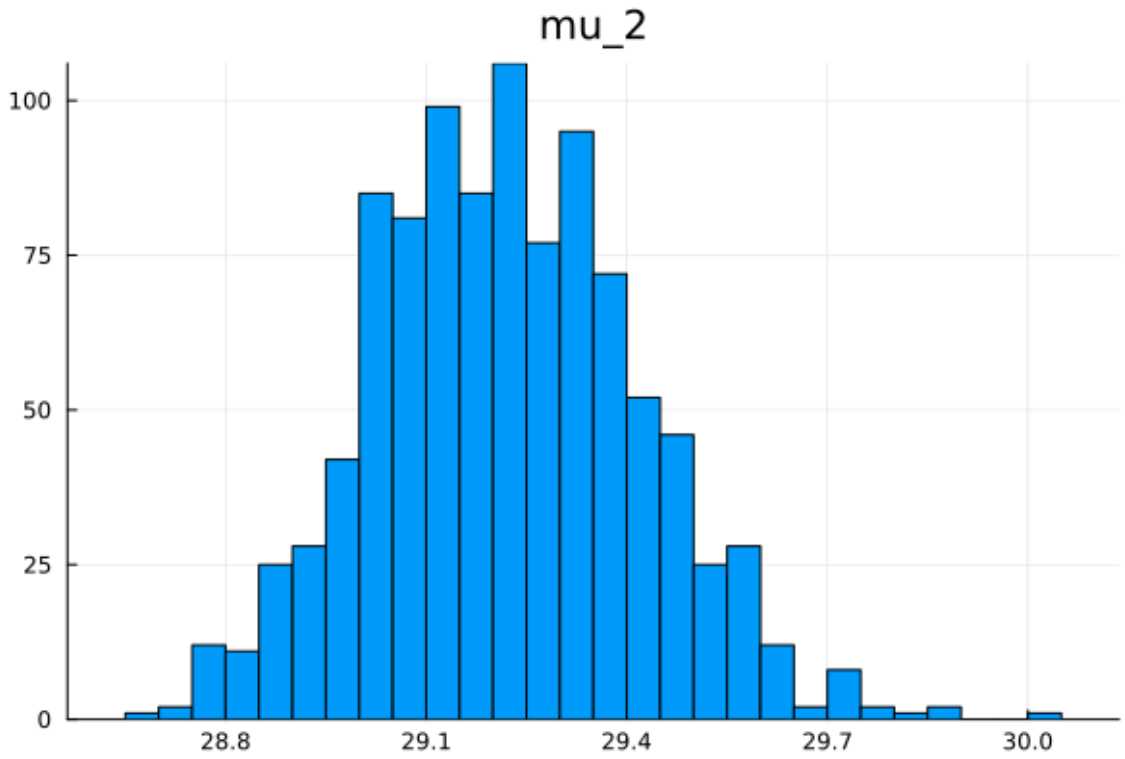
Out[49]:
```
(-11.084380005992855, 10.207454805686979)
```

Let's generate just the distribution of μ_2 and μ_n.

mu_1 = [] mu_2 = [] mu_n = [] n = 900 for i in 1:1000 M = rand_graph(n) e = eigvals(M) push!(mu_n, e[1]) push!(mu_2, e[n-1]) push!(mu_1, e[n]) end

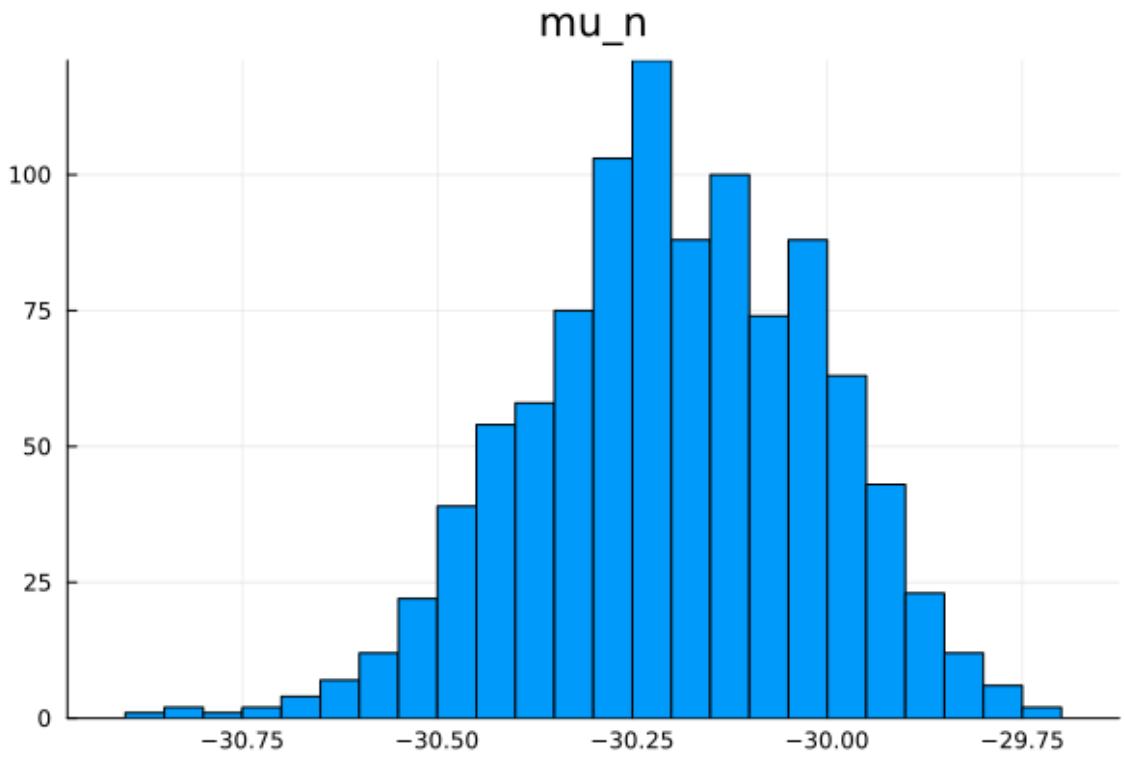
In [50]:
```
histogram(mu_2, legend=false, title="mu_2")
```

Out[50]:



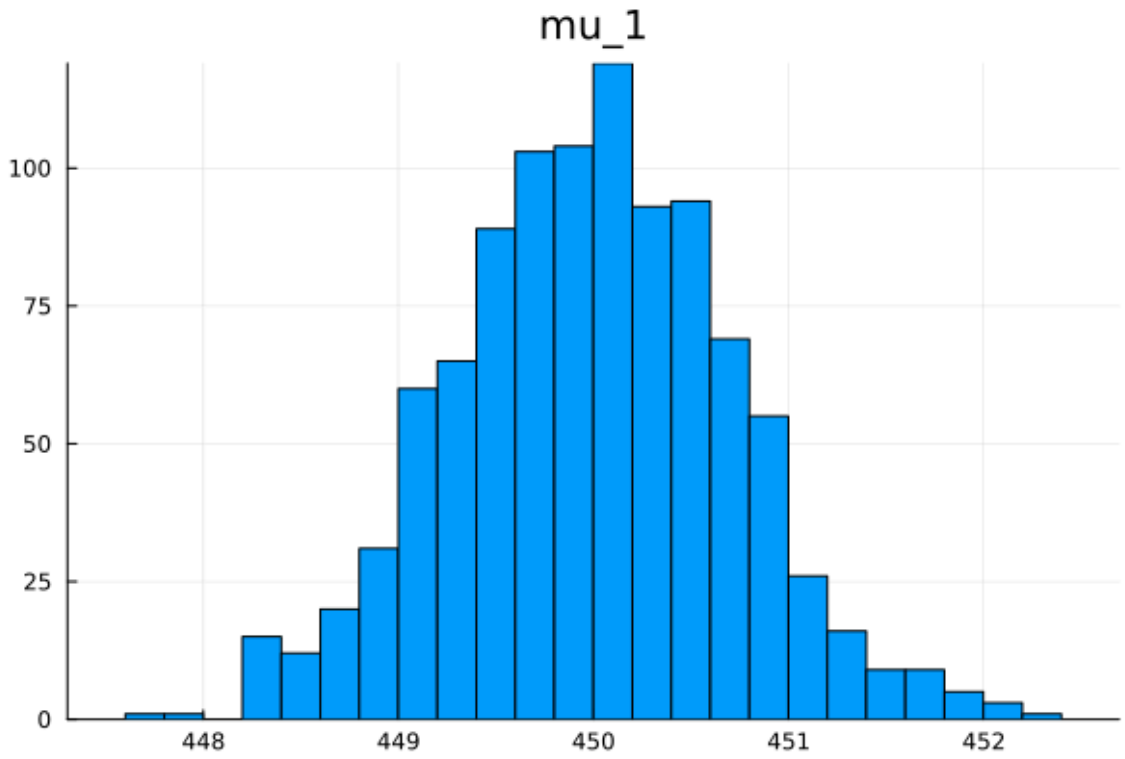
In [30]:
```
histogram(mu_n, legend=false, title="mu_n")
```

Out[30]:



In [51]:
```
histogram(mu_1, legend=false, title="mu_1")
```

Out[51]:



With p < 1/2

In [20]:
```
function rand_graph(n, p)
    M = triu(1 .* (rand(n,n) .< p) ,1)
    M = M + M'
end
rand_graph(10,0.2)
```

Out[20]:
```
10×10 Matrix{Int64}:
 0  0  0  0  0  0  0  0  0  0
 0  0  0  0  1  0  0  1  0  0
 0  0  0  0  0  0  0  0  1  1
 0  0  0  0  0  1  0  0  1  0
 0  1  0  0  0  0  0  0  0  0
 0  0  0  1  0  0  0  0  0  0
 0  0  0  0  0  0  0  1  0  0
 0  1  0  0  0  0  1  0  0  0
 0  0  1  1  0  0  0  0  0  1
 0  0  1  0  0  0  0  0  1  0
```

In [21]:
```
n = 1000
p = 0.1
M = rand_graph(1000, p)
    e = eigvals(M)
pp = p*(1-p)
minimum(e), e[end-1], 2*sqrt(p*(1-p)*n)
```

Out[21]:
```
(-19.18412008134872, 18.90027528407454, 18.973665961010276)
```

In [ ]: