10 Big Entropy and Generalized Linear Model

```julia
versioninfo()
```

```
Julia Version 1.10.2
Commit bd47eca2c8a (2024-03-01 10:14 UTC)
Build Info:
  Official https://julialang.org/ release
Platform Info:
  OS: Linux (x86_64-linux-gnu)
  CPU: 32 × Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-15.0.7 (ORCJIT, haswell)
Threads: 16 default, 0 interactive, 8 GC (on 32 virtua
l cores)
Environment:
  JULIA_PKG_SERVER = https://mirrors.tuna.tsinghua.ed
u.cn/julia
  JULIA_REVISE_WORKER_ONLY = 1
```

```julia
html"""<style>
main {
    margin: 0 auto;
    max-width: 90%;
    padding-left: max(50px, 1%);
    padding-right: max(253px, 10%);
    # 253px to accomodate TableOfContents(aside=true)
}
"""

```

```julia
using Pkg, DrWatson, PlutoUI
```

```julia
begin
    using DataFrames
    using StatsBase
    using StatsPlots
    using Random
    using Distributions
end
```

```julia
Plots.default(label=false);
```

Table of Contents

```
PlutoUI.TableOfContents()
```

10.1 Maximum entropy.

Code 10.1 Five hypothetical （A-E） distributions tossing 10 coins into 5 buckets

```
md"## Code 10.1 Five hypothetical (A-E) distributions
tossing 10 coins into 5 buckets"
```

p =

	A	B	C	D	E
1	0	0	0	1	2
2	0	1	2	2	2
3	10	8	6	4	2
4	0	1	2	2	2
5	0	0	0	1	2

```
p = DataFrame(
    :A => [0, 0, 10, 0, 0],
    :B => [0, 1, 8, 1, 0],
    :C => [0, 2, 6, 2, 0],
    :D => [1, 2, 4, 2, 1],
    :E => [2, 2, 2, 2, 2],
)
```

Code 10.2 Normalize the distributions

p_norm =

	A	B	C	D	E
1	0.0	0.0	0.0	0.1	0.2
2	0.0	0.1	0.2	0.2	0.2
3	1.0	0.8	0.6	0.4	0.2
4	0.0	0.1	0.2	0.2	0.2
5	0.0	0.0	0.0	0.1	0.2

```
p_norm = mapcols(c -> c ./ sum(c), p)
```

Code 10.3 Entropy for these 5 distributions

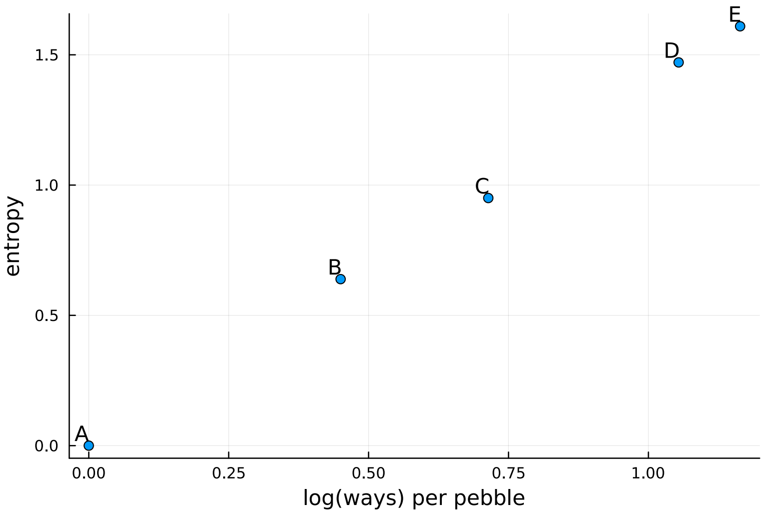
```
md" ## Code 10.3 Entropy for these 5 distributions"
```

ent_vals =

	A	B	C	D	E
1	-0.0	0.639032	0.950271	1.47081	1.60944

```
ent_vals = mapcols(entropy, p_norm)
```

Code 10.4, Figure 10.1



```
begin
    ways = [1, 90, 1260, 37800, 113400]
    logwayspp = log.(ways)/10

    txt = text.(names(ent_vals), :bottom, :right, 11)
    scatter(logwayspp, collect(ent_vals[1,:]), txt=txt,
    xlab="log(ways) per pebble", ylab="entropy")
end
```

Code 10.5 Calculate the expected number of blue marbles for all candidate distributions

- All must be equal to 1.

```
md" ## Code 10.5 Calculate the expected number of blue
marbles for all candidate distributions
- All must be equal to 1."
```

```
p2 =
▸[[0.25, 0.25, 0.25, 0.25], [0.333333, 0.166667, 0.166667, 0.3333
```

```
p2 = [
    [1/4, 1/4, 1/4, 1/4],
    [2/6, 1/6, 1/6, 2/6],
    [1/6, 2/6, 2/6, 1/6],
    [1/8, 4/8, 2/8, 1/8],
]
```

```
▸[1.0, 1.0, 1.0, 1.0]
```

```
map(x -> sum(x .* [0, 1, 1, 2]), p2)
```

Code 10.6 Calculate the entropy of each candidate distribution

```
md" ## Code 10.6 Calculate the entropy of each candidate distribution"
```

```
▸[1.38629, 1.32966, 1.32966, 1.21301]
```

```
# Could be simplified with just `map(entropy, p)`
# compute entropy of each distribution
map(x -> -sum(x .* log.(x)), p2)
```

Code 10.7 Binomial(2, 0.7): the expected number of blue marbles over 2 draws is 1.4.

- Blue vs white = 1.4 vs 0.6 = 7 vs 3
- Calculate the binomial prob for each 2-draw combo:
- WW, WB, BW, BB

```
md" ## Code 10.7 Binomial(2, 0.7): the expected number of blue marbles over 2 draws is 1.4.

- Blue vs white = 1.4 vs 0.6 = 7 vs 3
- Calculate the binomial prob for each 2-draw combo:
- WW, WB, BW, BB"

```

```
▸[0.09, 0.21, 0.21, 0.49]
```

```
begin
    p3 = 0.7
    A = [
        (1-p3)^2, p3*(1-p3), (1-p3)*p3, p3^2,
    ]
end
```

Code 10.8: entropy of the Binomial in 10.7

```
md" ## Code 10.8: entropy of the Binomial in 10.7"
```

```
1.221728604109787
```

```
-sum(A.*log.(A))
```

Code 10.9 sim_p: Simulate a random distribution with the same E(N*p)=1.4, N=2,p=0.7

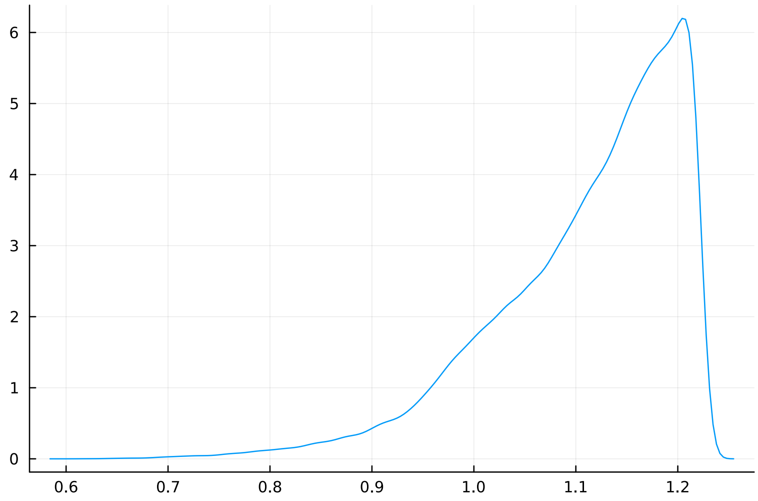
```
md" ## Code 10.9 sim_p: Simulate a random distribution with the same E(N*p)=1.4, N=2,p=0.7"
```

```
sim_p (generic function with 2 methods)
```

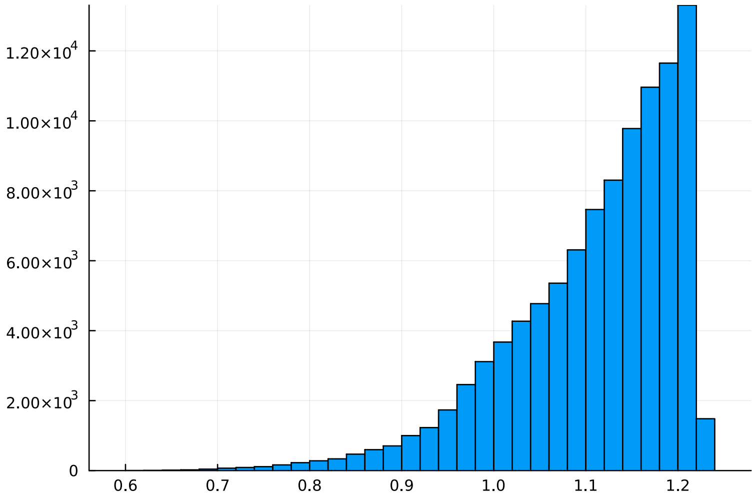
```julia
function sim_p(G::Float64 = 1.4)
    p = rand(Uniform(), 3)
    x4 = (G * sum(p) - p[2] - p[3])/(2-G)
    push!(p, x4)
    p ./= sum(p)
    (entropy(p), p)
end
```

Code 10.10 Histogram of entropy of all simulated distribution with N*p=1.4

```julia
md" ## Code 10.10 Histogram of entropy of all simulated
distribution with N*p=1.4"
```



```julia
begin
    Random.seed!(1)
    cnt = 10^5
    H = [sim_p(1.4) for _ ∈ 1:cnt];
    density(first.(H))
end
```



```julia
histogram(first.(H), bins=30)
```

Code 10.11 Fetch entropies and distributions

```
begin
    entropies = first.(H)
    distributions = last.(H);
end;
```

Code 10.12 Max entropy

```
1.221728429113852
```

```
maximum(entropies)
```

Code 10.13 The random distribution with the max entropy.

- It is almost identical to Binomial(2, 0.7).

```
md" ## Code 10.13 The random distribution with the max
entropy.
- It is almost identical to Binomial(2, 0.7)."
```

```
[0.0898778, 0.210088, 0.210157, 0.489878]
```

```
distributions[findmax(entropies)[2]]
```

10.2 Generalized linear models.

```
md" # 10.2 Generalized linear models."
```

```
# No code here, wheee!
```