

検定と信頼区間: 比率の検定と信頼区間

- 黒木玄
- 2022-05-31~2022-06-08

このノートでは[Julia言語 \(https://julialang.org/\)](https://julialang.org/)を使用している:

- [Julia言語のインストールの仕方の一例 \(https://nbviewer.org/github/genkuroki/msfd28/blob/master/install.ipynb\)](https://nbviewer.org/github/genkuroki/msfd28/blob/master/install.ipynb)

自明な誤りを見つけたら、自分で訂正して読んで欲しい。大文字と小文字の混同や書き直しが不完全な場合や符号のミスは非常によくある。

このノートに書いてある式を文字通りにそのまま読んで正しいと思ってしまうとひどい目に会う可能性が高い。しかし、数が使われている文献には大抵の場合に文字通りに読むと間違っている式や主張が書いてあるので、内容を理解した上で訂正しながら読んで利用しなければいけない。実践的に数学を使う状況では他人が書いた式をそのまま信じていけない。

このノートの内容よりもさらに詳しいノートを自分で作ると勉強になるだろう。膨大な時間を取られることになるが、このノートの内容に関係することで飯を食っていく可能性がある人にはそのためにかけた時間は無駄にならないと思われる。

目次

- ▼ [1 比率の検定と信頼区間](#)
 - [1.1 P値と信頼区間の定義の復習](#)
 - [1.2 二項分布モデルのP値達の定義を1枚の図で要約](#)
 - [1.3 4種のP値函数のプロット](#)
- ▼ [2 Clopper-Pearson](#)
 - [2.1 Clopper-Pearsonの信頼区間を与えるP値関数](#)
 - [2.2 Clopper-Pearsonの信頼区間](#)
 - [2.3 Clopper-Pearsonの信頼区間の図による把握](#)
- ▼ [3 Sterne](#)
 - [3.1 Sterneの信頼区間を与えるP値関数](#)
 - [3.2 Sterneの信頼区間](#)
 - [3.3 Sterneの信頼区間の図による把握](#)
 - [3.4 おまけ: R言語のbinom.testについて](#)
- ▼ [4 Wilson](#)
 - [4.1 Wilsonの信頼区間を与えるP値函数](#)
 - [4.2 Wilsonの信頼区間](#)
 - [4.3 Wilsonの信頼区間の図による把握](#)
 - [4.4 おまけ: R言語のprop.testについて](#)
 - [4.5 Pearsonの \$\chi^2\$ 検定との関係](#)
 - [4.6 おまけ: 多項分布モデルにおけるPearsonの \$\chi^2\$ 検定](#)
 - [4.7 おまけ: 一般の \$\chi^2\$ 検定で使う \$\chi^2\$ 分布の自由度](#)
- ▼ [5 Wald](#)
 - [5.1 Waldの信頼区間を与えるP値関数](#)
 - [5.2 Waldの信頼区間](#)
 - [5.3 Waldの信頼区間の図による把握](#)

```
In [1]: 1 ENV["LINES"], ENV["COLUMNS"] = 100, 100
2 using BenchmarkTools
3 using Distributions
4 using LinearAlgebra
5 using Printf
6 using QuadGK
7 using RCall
8 using Random
9 Random.seed!(4649373)
10 using Roots
11 using SpecialFunctions
12 using StaticArrays
13 using StatsBase
14 using StatsFuns
15 using StatsPlots
16 default(fmt = :png, size = (400, 250),
17         titlefontsize = 10, plot_titlefontsize = 12)
18 using SymPy
```

```
In [2]: 1 # Override the Base.show definition of SymPy.jl:
2 # https://github.com/JuliaPy/SymPy.jl/blob/29c5bfd1d10ac53014fa7fef468bc8deccadc2fc/src/types.jl
3
4 @eval SymPy function Base.show(io::IO, ::MIME"text/latex", x::SymbolicObject)
5     print(io, as_markdown("\displaystyle " * sympy.latex(x, mode="plain", fold_short_frac=false)))
6 end
7 @eval SymPy function Base.show(io::IO, ::MIME"text/latex", x::AbstractArray{Sym})
8     function toeqnarray(x::Vector{Sym})
9         a = join(["\displaystyle " * sympy.latex(x[i]) for i in 1:length(x)], "\\")
10        """\left[ \begin{array}{r}$a\end{array} \right]"""
11    end
12    function toeqnarray(x::AbstractArray{Sym,2})
13        sz = size(x)
14        a = join([join("\displaystyle " .* map(sympy.latex, x[i,:]), "&") for i in 1:sz[1]],
15                "\left[ \begin{array}{r}" * repeat("r",sz[2]) * "}" * a * "\end{array}\right]")
16    end
17    print(io, as_markdown(toeqnarray(x)))
18 end
```

```
In [3]: 1 safemul(x, y) = x == 0 ? x : x*y
2 safediv(x, y) = x == 0 ? x : x/y
3
4 x ≲ y = x < y || x ≈ y
5
6 mypdf(dist, x) = pdf(dist, x)
7 mypdf(dist::DiscreteUnivariateDistribution, x) = pdf(dist, round(Int, x))
8
9 distname(dist::Distribution) = replace(string(dist), r"{.*}" => "")
10 myskewness(dist) = skewness(dist)
11 mykurtosis(dist) = kurtosis(dist)
12 function standardized_moment(dist::ContinuousUnivariateDistribution, m)
13     μ, σ = mean(dist), std(dist)
14     quadgk(x → (x - μ)^m * pdf(dist, x), extrema(dist)...)[1] / σ^m
15 end
16 myskewness(dist::MixtureModel{Univariate, Continuous}) = standardized_moment(dist, 3)
17 mykurtosis(dist::MixtureModel{Univariate, Continuous}) = standardized_moment(dist, 4) - 3
```

Out[3]: mykurtosis (generic function with 2 methods)

Clopper-Pearsonの節の準備

```
In [4]: 1 function pvalue_clopper_pearson(dist::DiscreteUnivariateDistribution, x)
2     min(1, 2cdf(dist, x), 2ccdf(dist, x-1))
3 end
4 pvalue_clopper_pearson(n, k, p) = pvalue_clopper_pearson(Binomial(n, p), k)
5
6 function confint_clopper_pearson(n, k; α = 0.05)
7     p_L = k > 0 ? quantile(Beta(k, n-k+1), α/2) : zero(α)
8     p_U = k < n ? quantile(Beta(k+1, n-k), 1-α/2) : one(α)
9     [p_L, p_U]
10 end
```

Out[4]: confint_clopper_pearson (generic function with 1 method)

```
In [5]: 1 n = 10
        2 k = 0:n
        3 [
        4     k/n;;
        5     vcat(confint_clopper_pearson.(n, k)'...);;
        6 ]
```

```
Out[5]: 11×3 Matrix{Float64}:
 0.0  0.0      0.308497
 0.1  0.00252858 0.445016
 0.2  0.0252107  0.556095
 0.3  0.0667395  0.652453
 0.4  0.121552   0.737622
 0.5  0.187086   0.812914
 0.6  0.262378   0.878448
 0.7  0.347547   0.93326
 0.8  0.443905   0.974789
 0.9  0.554984   0.997471
 1.0  0.691503   1.0
```

```
In [6]: 1 n, k, α = 100, 30, 0.05
        2 kmin, kmax = 0, 65
        3 p_L, p_U = confint_clopper_pearson(n, k; α)
        4 @show [p_L, p_U];

[p_L, p_U] = [0.21240642048953662, 0.39981467617980404]
```

```
In [7]: 1 dist_L, dist_U = Binomial(n, p_L), Binomial(n, p_U)
```

```
Out[7]: (Binomial{Float64}(n=100, p=0.21240642048953662), Binomial{Float64}(n=100, p=0.39981467617980404))
```

```
In [8]: 1 var"Clopper-Pearson: 下側では p = p_L がぎりぎり" = plot(; size=(500, 300))
        2 plot!(k → mypdf(dist_L, k), kmin-0.5, kmax+0.5;
        3     label="", c=1)
        4 plot!(k → mypdf(dist_U, k), k-0.5, kmax+0.5;
        5     fillrange=0, c=1, fc=:blue, fa=0.5, label="")
        6 annotate!(k+2, 0.01, text("$ (100α/2)%", :blue, :left, 10))
        7 title!("Binomial(n=$n, p=p_L), p_L=$(round(p_L; digits=4))")
        8 vline!([k]; label="k=$k", c=:black, ls=:dot)
        9 vline!([n*p_L]; label="n p_L = $(round(n*p_L; digits=2))", c=:blue, ls=:dash)
        10 plot!(); ylim=(-0.007, 0.105));
```

```
In [9]: 1 var"Clopper-Pearson: 上側では p = p_U がぎりぎり" = plot(; size=(500, 300))
        2 plot!(k → mypdf(dist_U, k), kmin-0.5, kmax+0.5;
        3     label="", c=2)
        4 plot!(k → mypdf(dist_L, k), kmin-0.5, k+0.5;
        5     fillrange=0, c=2, fc=:red, fa=0.5, label="")
        6 annotate!(k-2, 0.01, text("$ (100α/2)%", :red, :right, 10))
        7 title!("Binomial(n=$n, p=p_U), p_U=$(round(p_U; digits=4))")
        8 vline!([k]; label="k=$k", c=:black, ls=:dot)
        9 vline!([n*p_U]; label="n p_U = $(round(n*p_U; digits=2))", c=:red, ls=:dashdot)
        10 plot!(); ylim=(-0.007, 0.105));
```

```

In [10]: 1 var"Clopper-Pearson: n × 95%信頼区間" = plot(; size=(500, 300))
2
3 plot!(k → mypdf(dist_L, k), kmin-0.5, 47+0.5;
4       label="", c=1)
5 plot!(k → mypdf(dist_L, k), k-0.5, 47+0.5;
6       fillrange=0, c=1, fc=:blue, fa=0.5, label="")
7 annotate!(k+2, 0.01, text("$(100α/2)%", :blue, :left, 10))
8
9 plot!(k → mypdf(dist_U, k), 15-0.5, kmax+0.5;
10      label="", c=2)
11 plot!(k → mypdf(dist_U, k), 15-0.5, k+0.5;
12      fillrange=0, c=2, fc=:red, fa=0.5, label="")
13 annotate!(k-2, 0.01, text("$(100α/2)%", :red, :right, 10))
14
15 vline!([k]; label="k=$k", c=:black, ls=:dot)
16 vline!([n*p_L]; label="n p_L = $(round(n*p_L; digits=2))", c=:blue, ls=:dash)
17 vline!([n*p_U]; label="n p_U = $(round(n*p_U; digits=2))", c=:red, ls=:dashdot)
18 plot!([n*p_L, n*p_U], fill(-0.003, 2); label="", c=:red, lw=5)
19 title!("n = $n, [n p_L, n p_U] = [$(round(n*p_L; digits=2)), $(round(n*p_U; digits=2))]")
20 plot!(); ylim=(-0.007, 0.105));

```

Sterneの節の準備

```

In [11]: 1 _pdf_le(x, (dist, y)) = pdf(dist, x) ≤ y
2
3 function _search_boundary(f, x0, Δx, param)
4     x = x0
5     if f(x, param)
6         while f(x - Δx, param) x -= Δx end
7     else
8         x += Δx
9         while !f(x, param) x += Δx end
10    end
11    x
12 end
13
14 function pvalue_sterne(dist::DiscreteUnivariateDistribution, x)
15     Px = pdf(dist, x)
16     Px == 0 && return Px
17     Px == 1 && return Px
18     m = mode(dist)
19     Px ≈ pdf(dist, m) && return one(Px)
20     if x < m
21         y = _search_boundary(_pdf_le, 2m - x, 1, (dist, Px))
22         cdf(dist, x) + ccdf(dist, y-1)
23     else # x > m
24         y = _search_boundary(_pdf_le, 2m - x, -1, (dist, Px))
25         cdf(dist, y) + ccdf(dist, x-1)
26     end
27 end
28
29 function pvalue_sterne(n, k, p)
30     0 < p ≤ 1 ? pvalue_sterne(Binomial(n, p), k) : zero(p)
31 end
32
33 function confint_sterne(n, k; α = 0.05)
34     find_zeros(p → log(pvalue_sterne(n, k, p)) - log(α), -1.01, 1.01)
35 end

```

Out[11]: confint_sterne (generic function with 1 method)

```
In [12]: 1 n = 10
2 k = 0:n
3 [
4     k/n;;
5     vcat(confint_clopper_pearson.(n, k)'...);;
6     vcat(confint_sterne.(n, k)'...);;
7 ]
```

```
Out[12]: 11×5 Matrix{Float64}:
0.0  0.0      0.308497  0.0      0.290865
0.1  0.00252858 0.445016  0.0051162 0.446489
0.2  0.0252107  0.556095  0.0367714 0.553511
0.3  0.0667395  0.652453  0.0872644 0.619411
0.4  0.121552   0.737622  0.150028  0.709135
0.5  0.187086   0.812914  0.222441  0.777559
0.6  0.262378   0.878448  0.290865  0.849972
0.7  0.347547   0.93326   0.380589  0.912736
0.8  0.443905   0.974789  0.446489  0.963229
0.9  0.554984   0.997471  0.553511  0.994884
1.0  0.691503   1.0      0.709135  1.0
```

```
In [13]: 1 n, k, α = 100, 30, 0.05
2 kmin, kmax = 0, 65
3 p_L, p_U = confint_sterne(n, k; α)
4 @show [p_L, p_U];
```

```
[p_L, p_U] = [0.21419730401454778, 0.3996107372232876]
```

```
In [14]: 1 dist_L = Binomial(n, p_L)
2 k_L = minimum(i for i in 0:n if !(pdf(dist_L, i) ≤ pdf(dist_L, k))) - 1
3 cdf_L = cdf(dist_L, k_L)
4 ccdf_L = ccdf(dist_L, k-1)
5 cdf_L + ccdf_L
```

```
Out[14]: 0.04999999999999993
```

```
In [15]: 1 dist_U = Binomial(n, p_U)
2 k_U = maximum(i for i in 0:n if !(pdf(dist_U, i) ≤ pdf(dist_U, k))) + 1
3 cdf_U = cdf(dist_U, k)
4 ccdf_U = ccdf(dist_U, k_U-1)
5 cdf_U + ccdf_U
```

```
Out[15]: 0.05184087343317326
```

```
In [16]: 1 var"Sterne: 下側では p = p_L がぎりぎり" = plot(; size=(500, 300))
2 plot!(i → mypdf(dist_L, i), kmin-0.5, kmax+0.5;
3     label="", c=1)
4 plot!(i → mypdf(dist_L, i), k-0.5, kmax+0.5;
5     fillrange=0, c=1, fc=:blue, fa=0.5, label="")
6 plot!(i → mypdf(dist_L, i), kmin-0.5, k_L+0.5;
7     fillrange=0, c=1, fc=:blue, fa=0.5, label="")
8 plot!([k_L+0.5, k-0.5], fill(pdf(dist_L, k), 2); label="", c=:blue, ls=:dot)
9 annotate!(k+2, 0.01, text("$(round(100ccdf_L; digits=1))%", :blue, :left, 10))
10 annotate!(k_L-2, 0.01, text("$(round(100cdf_L; digits=1))%", :blue, :right, 10))
11 title!("Binomial(n=$n, p=p_L), p_L=$(round(p_L; digits=4))")
12 vline!([k]; label="k=$k", c=:black, ls=:dot)
13 vline!([n*p_L]; label="n p_L = $(round(n*p_L; digits=2))", c=:blue, ls=:dash)
14 plot!(); ylim=(-0.007, 0.105));
```

```
In [17]: 1 var"Sterne: 上側では p = p_U がぎりぎり" = plot(; size=(500, 300))
2 plot!(i → mypdf(dist_U, i), kmin-0.5, kmax+0.5;
3     label="", c=2)
4 plot!(i → mypdf(dist_U, i), kmin-0.5, k+0.5;
5     fillrange=0, c=2, fc=:red, fa=0.5, label="")
6 plot!(i → mypdf(dist_U, i), k_U-0.5, kmax+0.5;
7     fillrange=0, c=2, fc=:red, fa=0.5, label="")
8 plot!([k+0.5, k_U-0.5], fill(pdf(dist_U, k), 2); label="", c=:red, ls=:dot)
9 annotate!(k_U+2, 0.01, text("$(round(100ccdf_U; digits=1))%", :red, :left, 10))
10 annotate!(k-2, 0.01, text("$(round(100cdf_U; digits=1))%", :red, :right, 10))
11 title!("Binomial(n=$n, p=p_U), p_U=$(round(p_U; digits=4))")
12 vline!([k]; label="k=$k", c=:black, ls=:dot)
13 vline!([n*p_U]; label="n p_U = $(round(n*p_U; digits=2))", c=:red, ls=:dashdot)
14 plot!(); ylim=(-0.007, 0.105));
```

```
In [18]: 1 var"Sterne: n × 95%信頼区間" = plot(; size=(500, 300))
2
3 plot!(k → mypdf(dist_L, k), kmin-0.5, 47+0.5;
4       label="", c=1)
5 plot!(k → mypdf(dist_L, k), k-0.5, 47+0.5;
6       fillrange=0, c=1, fc=:blue, fa=0.5, label="")
7 plot!(i → mypdf(dist_L, i), kmin-0.5, k_L+0.5;
8       fillrange=0, c=1, fc=:blue, fa=0.5, label="")
9 #plot!([k_L+0.5, k-0.5], fill(pdf(dist_L, k), 2); label="", c=:blue, ls=:dot)
10 annotate!(k+2, 0.01, text("$(round(100ccdf_L; digits=1))%", :blue, :left, 10))
11 annotate!(k_L-2, 0.01, text("$(round(100ccdf_L; digits=1))%", :blue, :right, 10))
12
13 plot!(i → mypdf(dist_U, i), 15-0.5, kmax+0.5;
14       label="", c=2)
15 plot!(i → mypdf(dist_U, i), 15-0.5, k+0.5;
16       fillrange=0, c=2, fc=:red, fa=0.5, label="")
17 plot!(i → mypdf(dist_U, i), k_U-0.5, kmax+0.5;
18       fillrange=0, c=2, fc=:red, fa=0.5, label="")
19 #plot!([k+0.5, k_U-0.5], fill(pdf(dist_U, k), 2); label="", c=:red, ls=:dot)
20 annotate!(k_U+2, 0.01, text("$(round(100ccdf_U; digits=1))%", :red, :left, 10))
21 annotate!(k-2, 0.01, text("$(round(100ccdf_U; digits=1))%", :red, :right, 10))
22
23 vline!([k]; label="k=$k", c=:black, ls=:dot)
24 vline!([n*p_L]; label="n p_L = $(round(n*p_L; digits=2))", c=:blue, ls=:dash)
25 vline!([n*p_U]; label="n p_U = $(round(n*p_U; digits=2))", c=:red, ls=:dashdot)
26 plot!([n*p_L, n*p_U], fill(-0.003, 2); label="", c=:red, lw=5)
27 title!("n = $n, [n p_L, n p_U] = [$(round(n*p_L; digits=2)), $(round(n*p_U; digits=2))]")
28 plot!(); ylim=(-0.007, 0.105));
```

Wilsonの節の準備

```
In [19]: 1 function pvalue_wilson(n, k, p)
2     p̂ = k/n
3     SE = √(p*(1-p)/n)
4     2ccdf(Normal(), abs(p̂ - p)/SE)
5 end
6
7 function confint_wilson(n, k; α = 0.05)
8     p̂ = k/n
9     z = quantile(Normal(), 1-α/2)
10    a, b, c = 1+z^2/n, p̂+z^2/(2n), p̂^2
11    sqrtD = √(b^2 - a*c)
12    p_L = (b - sqrtD)/a
13    p_U = (b + sqrtD)/a
14    [p_L, p_U]
15 end
```

Out[19]: confint_wilson (generic function with 1 method)

```
In [20]: 1 n = 10
2 k = 0:n
3 [
4     k/n;;
5     vcat(confint_clopper_pearson.(n, k)'...);;
6     vcat(confint_sterne.(n, k)'...);;
7     vcat(confint_wilson.(n, k)'...);;
8 ]
```

Out[20]: 11×7 Matrix{Float64}:

0.0	0.0	0.308497	0.0	0.290865	0.0	0.277533
0.1	0.00252858	0.445016	0.0051162	0.446489	0.0178762	0.40415
0.2	0.0252107	0.556095	0.0367714	0.553511	0.0566822	0.509838
0.3	0.0667395	0.652453	0.0872644	0.619411	0.107791	0.603222
0.4	0.121552	0.737622	0.150028	0.709135	0.16818	0.687326
0.5	0.187086	0.812914	0.222441	0.777559	0.236593	0.763407
0.6	0.262378	0.878448	0.290865	0.849972	0.312674	0.83182
0.7	0.347547	0.93326	0.380589	0.912736	0.396778	0.892209
0.8	0.443905	0.974789	0.446489	0.963229	0.490162	0.943318
0.9	0.554984	0.997471	0.553511	0.994884	0.59585	0.982124
1.0	0.691503	1.0	0.709135	1.0	0.722467	1.0

```
In [21]: 1  $\alpha = 0.05$ 
2 z = quantile(Normal(), 1 -  $\alpha/2$ )
3
4 var"z_{0.025} の定義" = plot()
5 plot!(Normal(), -5, 5; label="")
6 plot!(Normal(), -5, -z; label="", c=1, fillrange=0, fc=:red, fa=0.3)
7 plot!(Normal(), z, 5; label="", c=1, fillrange=0, fc=:red, fa=0.3)
8 plot!([-z, -z, NaN, z, z],
9      [0, pdf(Normal(), -z), NaN, 0, pdf(Normal(), z)]);
10      label="\$\\pm z_{0.025}\$", c=:red)
11 title!("Normal(0,1)"; xtick=-10:10)
12 annotate!(-2.2, 0.06, text("2.5%", 10, :red, :right))
13 annotate!(2.2, 0.06, text("2.5%", 10, :red, :left))
14 plot!(); legendfontsize=14);
```

```
In [22]: 1 n, k,  $\alpha$  = 100, 30, 0.05
2 kmin, kmax = 0, 65
3 p_L, p_U = confint_wilson(n, k;  $\alpha$ )
4 @show [p_L, p_U];

[p_L, p_U] = [0.2189488529493274, 0.395848546333467]
```

```
In [23]: 1 dist_L, dist_U = Binomial(n, p_L), Binomial(n, p_U)
2 normal_L = Normal(n*p_L,  $\sqrt{n*p_L*(1-p_L)}$ )
3 normal_U = Normal(n*p_U,  $\sqrt{n*p_U*(1-p_U)}$ )
4 k_L, k_U = 2n*p_L - k, 2n*p_U - k
5 cdf_L, ccdf_L = cdf(normal_L, k_L), ccdf(normal_L, k)
6 cdf_U, ccdf_U = cdf(normal_U, k), ccdf(normal_U, k_U);
```

```
In [24]: 1 var"Wilson: 下側では p = p_L がぎりぎり" = plot(; size=(500, 300))
2 plot!(i  $\rightarrow$  mypdf(dist_L, i), kmin-0.5, kmax+0.5;
3      label="", c=1)
4 plot!(normal_L; label="normal approx", c=:blue)
5 plot!(x  $\rightarrow$  pdf(normal_L, x), k, kmax+0.5;
6      fillrange=0, c=:blue, fc=:blue, fa=0.5, label="", ls=:dot)
7 plot!(x  $\rightarrow$  pdf(normal_L, x), kmin-0.5, k_L;
8      fillrange=0, c=:blue, fc=:blue, fa=0.5, label="")
9 annotate!(k+2, 0.01, text("$(round(100ccdf_L; digits=1))\%", :blue, :left, 10))
10 annotate!(k_L-2, 0.01, text("$(round(100cdf_L; digits=1))\%", :blue, :right, 10))
11 title!("Binomial(n=$n, p=p_L), p_L=$(round(p_L; digits=4))")
12 vline!([k]; label="k=$k", c=:black, ls=:dot)
13 vline!([n*p_L]; label="n p_L = $(round(n*p_L; digits=2))", c=:blue, ls=:dash)
14 plot!(); ylim=(-0.007, 0.105));
```

```
In [25]: 1 var"Wilson: 上側では p = p_U がぎりぎり" = plot(; size=(500, 300))
2 plot!(i  $\rightarrow$  mypdf(dist_U, i), kmin-0.5, kmax+0.5;
3      label="", c=2)
4 plot!(normal_U; label="normal approx", c=:red)
5 plot!(x  $\rightarrow$  pdf(normal_U, x), k_U, kmax+0.5;
6      fillrange=0, c=:red, fc=:red, fa=0.5, label="", ls=:dot)
7 plot!(x  $\rightarrow$  pdf(normal_U, x), kmin-0.5, k;
8      fillrange=0, c=:red, fc=:red, fa=0.5, label="")
9 annotate!(k_U+2, 0.01, text("$(round(100ccdf_U; digits=1))\%", :red, :left, 10))
10 annotate!(k-2, 0.01, text("$(round(100cdf_U; digits=1))\%", :red, :right, 10))
11 title!("Binomial(n=$n, p=p_U), p_U=$(round(p_U; digits=4))")
12 vline!([k]; label="k=$k", c=:black, ls=:dot)
13 vline!([n*p_U]; label="n p_U = $(round(n*p_U; digits=2))", c=:red, ls=:dashdot)
14 plot!(); ylim=(-0.007, 0.105));
```



```
In [26]: 1 var"Wilson: n × 95%信頼区間" = plot(; size=(500, 300))
2
3 plot!(i → mypdf(dist_L, i), kmin-0.5, 47+0.5;
4       label="", c=1)
5 plot!(normal_L; label="normal approx", c=:blue)
6 plot!(x → pdf(normal_L, x), k, 47+0.5;
7       fillrange=0, c=:blue, fc=:blue, fa=0.5, label="", ls=:dot)
8 plot!(x → pdf(normal_L, x), kmin-0.5, k_L;
9       fillrange=0, c=:blue, fc=:blue, fa=0.5, label="")
10 annotate!(k+2, 0.01, text("$(round(100ccdf_L; digits=1))%", :blue, :left, 10))
11 annotate!(k_L-2, 0.01, text("$(round(100ccdf_L; digits=1))%", :blue, :right, 10))
12
13 plot!(i → mypdf(dist_U, i), 15-0.5, kmax+0.5;
14       label="", c=2)
15 plot!(normal_U; label="normal approx", c=:red)
16 plot!(x → pdf(normal_U, x), k_U, kmax+0.5;
17       fillrange=0, c=:red, fc=:red, fa=0.5, label="", ls=:dot)
18 plot!(x → pdf(normal_U, x), 15-0.5, k;
19       fillrange=0, c=:red, fc=:red, fa=0.5, label="")
20 annotate!(k_U+2, 0.01, text("$(round(100ccdf_U; digits=1))%", :red, :left, 10))
21 annotate!(k-2, 0.01, text("$(round(100ccdf_U; digits=1))%", :red, :right, 10))
22
23 vline!([k]; label="k=$k", c=:black, ls=:dot)
24 vline!([n*p_L]; label="n p_L = $(round(n*p_L; digits=2))", c=:blue, ls=:dash)
25 vline!([n*p_U]; label="n p_U = $(round(n*p_U; digits=2))", c=:red, ls=:dashdot)
26 plot!([n*p_L, n*p_U], fill(-0.003, 2); label="", c=:red, lw=5)
27 title!("n = $n, [n p_L, n p_U] = [$(round(n*p_L; digits=2)), $(round(n*p_U; digits=2))]")
28 plot!(); ylim=(-0.007, 0.105));
```

Waldの節の準備

```
In [27]: 1 function pvalue_wald(n, k, p)
2     p̂ = k/n
3     SEhat = √(p̂*(1-p̂)/n)
4     2ccdf(Normal(), abs(p̂ - p)/SEhat)
5 end
6
7 function confint_wald(n, k; α = 0.05)
8     p̂ = k/n
9     SEhat = √(p̂*(1-p̂)/n)
10    z = quantile(Normal(), 1-α/2)
11    p_L = p̂ - z*SEhat
12    p_U = p̂ + z*SEhat
13    [p_L, p_U]
14 end
```

Out[27]: confint_wald (generic function with 1 method)

```
In [28]: 1 n = 10
2 k = 0:n
3 [
4     k/n;;
5     vcat(confint_clopper_pearson.(n, k)'...);;
6     vcat(confint_sterne.(n, k)'...);;
7     vcat(confint_wilson.(n, k)'...);;
8     vcat(confint_wald.(n, k)'...);;
9 ]
```

Out[28]: 11×9 Matrix{Float64}:

0.0	0.0	0.308497	0.0	0.290865	0.0	0.277533	0.0	0.0
0.1	0.00252858	0.445016	0.0051162	0.446489	0.0178762	0.40415	-0.0859385	0.285939
0.2	0.0252107	0.556095	0.0367714	0.553511	0.0566822	0.509838	-0.047918	0.447918
0.3	0.0667395	0.652453	0.0872644	0.619411	0.107791	0.603222	0.0159742	0.584026
0.4	0.121552	0.737622	0.150028	0.709135	0.16818	0.687326	0.0963637	0.703636
0.5	0.187086	0.812914	0.222441	0.777559	0.236593	0.763407	0.190102	0.809898
0.6	0.262378	0.878448	0.290865	0.849972	0.312674	0.83182	0.296364	0.903636
0.7	0.347547	0.93326	0.380589	0.912736	0.396778	0.892209	0.415974	0.984026
0.8	0.443905	0.974789	0.446489	0.963229	0.490162	0.943318	0.552082	1.04792
0.9	0.554984	0.997471	0.553511	0.994884	0.59585	0.982124	0.714061	1.08594
1.0	0.691503	1.0	0.709135	1.0	0.722467	1.0	1.0	1.0


```
In [29]: 1 n = 100
2 k = 0:10:n
3 [
4     k/n;;
5     vcat(confint_clopper_pearson.(n, k)'...);;
6     vcat(confint_sterne.(n, k)'...);;
7     vcat(confint_wilson.(n, k)'...);;
8     vcat(confint_wald.(n, k)'...);;
9 ]
```

```
Out[29]: 11x9 Matrix{Float64}:
0.0 0.0 0.0362167 0.0 0.0375544 0.0 0.0369935 0.0 0.0
0.1 0.0490047 0.176223 0.0533912 0.174 0.0552291 0.174366 0.0412011 0.158799
0.2 0.126656 0.291843 0.128607 0.289343 0.133367 0.288829 0.121601 0.278399
0.3 0.212406 0.399815 0.214197 0.399611 0.218949 0.395849 0.210183 0.389817
0.4 0.303295 0.502791 0.305511 0.5 0.309401 0.497997 0.303982 0.496018
0.5 0.398321 0.601679 0.399611 0.600389 0.403832 0.596168 0.402002 0.597998
0.6 0.497209 0.696705 0.5 0.694489 0.502003 0.690599 0.503982 0.696018
0.7 0.600185 0.787594 0.600389 0.785803 0.604151 0.781051 0.610183 0.789817
0.8 0.708157 0.873344 0.710657 0.871393 0.711171 0.866633 0.721601 0.878399
0.9 0.823777 0.950995 0.826 0.946609 0.825634 0.944771 0.841201 0.958799
1.0 0.963783 1.0 0.962446 1.0 0.963007 1.0 1.0 1.0
```

```
In [30]: 1 n = 1000
2 k = 0:100:n
3 [
4     k/n;;
5     vcat(confint_clopper_pearson.(n, k)'...);;
6     vcat(confint_sterne.(n, k)'...);;
7     vcat(confint_wilson.(n, k)'...);;
8     vcat(confint_wald.(n, k)'...);;
9 ]
```

```
Out[30]: 11x9 Matrix{Float64}:
0.0 0.0 0.00368208 0.0 0.00376337 0.0 0.00382676 0.0 0.0
0.1 0.0821053 0.120288 0.0823528 0.120377 0.0829094 0.120152 0.0814061 0.118594
0.2 0.175621 0.226159 0.175895 0.225921 0.176377 0.225919 0.175208 0.224792
0.3 0.271721 0.329462 0.271927 0.329447 0.272407 0.329124 0.271597 0.328403
0.4 0.369469 0.431122 0.369963 0.430979 0.370075 0.430691 0.369636 0.430364
0.5 0.468549 0.531451 0.468991 0.531009 0.46907 0.53093 0.46901 0.53099
0.6 0.568878 0.630531 0.569021 0.630037 0.569309 0.629925 0.569636 0.630364
0.7 0.670538 0.728279 0.670553 0.728073 0.670876 0.727593 0.671597 0.728403
0.8 0.773841 0.824379 0.774079 0.824105 0.774081 0.823623 0.775208 0.824792
0.9 0.879712 0.917895 0.879623 0.917647 0.879848 0.917091 0.881406 0.918594
1.0 0.996318 1.0 0.996237 1.0 0.996173 1.0 1.0 1.0
```

```
In [31]: 1 n, k, α = 100, 30, 0.05
2 kmin, kmax = 0, 65
3 p_L, p_U = confint_wald(n, k; α)
4 @show [p_L, p_U];
```

```
[p_L, p_U] = [0.21018316681457927, 0.3898168331854207]
```

```
In [32]: 1 dist_L, dist_U = Binomial(n, p_L), Binomial(n, p_U)
2 p̂ = k/n
3 normal_L = Normal(n*p_L, √(n*p̂*(1-p̂)))
4 normal_U = Normal(n*p_U, √(n*p̂*(1-p̂)))
5 k_L, k_U = 2n*p_L - k, 2n*p_U - k
6 cdf_L, ccdf_L = cdf(normal_L, k_L), ccdf(normal_L, k)
7 cdf_U, ccdf_U = cdf(normal_U, k), ccdf(normal_U, k_U);
```

```
In [33]: 1 var"Wald: 下側では p = p_L がぎりぎり" = plot(; size=(500, 300))
2 plot!(i → mypdf(dist_L, i), kmin-0.5, kmax+0.5;
3       label="", c=1)
4 plot!(normal_L; label="normal approx", c=:blue)
5 plot!(x → pdf(normal_L, x), k, kmax+0.5;
6       fillrange=0, c=:blue, fc=:blue, fa=0.5, label="", ls=:dot)
7 plot!(x → pdf(normal_L, x), kmin-0.5, k_L;
8       fillrange=0, c=:blue, fc=:blue, fa=0.5, label="")
9 annotate!(k+2, 0.01, text("$(round(100ccdf_L; digits=1))%", :blue, :left, 10))
10 annotate!(k_L-2, 0.01, text("$(round(100cdf_L; digits=1))%", :blue, :right, 10))
11 title!("Binomial(n=$n, p=p_L), p_L=$(round(p_L; digits=4))")
12 vline!([k]; label="k=$k", c=:black, ls=:dot)
13 vline!([n*p_L]; label="n p_L = $(round(n*p_L; digits=2))", c=:blue, ls=:dash)
14 plot!(); ylim=(-0.007, 0.105));
```

```
In [34]: 1 var"Wald: 上側では p = p_U がぎりぎり" = plot(; size=(500, 300))
2 plot!(i → mypdf(dist_U, i), kmin-0.5, kmax+0.5;
3       label="", c=2)
4 plot!(normal_U; label="normal approx", c=:red)
5 plot!(x → pdf(normal_U, x), k_U, kmax+0.5;
6       fillrange=0, c=:red, fc=:red, fa=0.5, label="", ls=:dot)
7 plot!(x → pdf(normal_U, x), kmin-0.5, k;
8       fillrange=0, c=:red, fc=:red, fa=0.5, label="")
9 annotate!(k_U+2, 0.01, text("$(round(100ccdf_U; digits=1))%", :red, :left, 10))
10 annotate!(k-2, 0.01, text("$(round(100cdf_U; digits=1))%", :red, :right, 10))
11 title!("Binomial(n=$n, p=p_U), p_U=$(round(p_U; digits=4))")
12 vline!([k]; label="k=$k", c=:black, ls=:dot)
13 vline!([n*p_U]; label="n p_U = $(round(n*p_U; digits=2))", c=:red, ls=:dashdot)
14 plot!(); ylim=(-0.007, 0.105));
```

```
In [35]: 1 var"Wald: n × 95%信頼区間" = plot(; size=(500, 300))
2
3 plot!(i → mypdf(dist_L, i), kmin-0.5, 47+0.5;
4       label="", c=1)
5 plot!(normal_L; label="normal approx", c=:blue)
6 plot!(x → pdf(normal_L, x), k, 47+0.5;
7       fillrange=0, c=:blue, fc=:blue, fa=0.5, label="", ls=:dot)
8 plot!(x → pdf(normal_L, x), kmin-0.5, k_L;
9       fillrange=0, c=:blue, fc=:blue, fa=0.5, label="")
10 annotate!(k+2, 0.01, text("$(round(100ccdf_L; digits=1))%", :blue, :left, 10))
11 annotate!(k_L-2, 0.01, text("$(round(100cdf_L; digits=1))%", :blue, :right, 10))
12
13 plot!(i → mypdf(dist_U, i), 15-0.5, kmax+0.5;
14       label="", c=2)
15 plot!(normal_U; label="normal approx", c=:red)
16 plot!(x → pdf(normal_U, x), k_U, kmax+0.5;
17       fillrange=0, c=:red, fc=:red, fa=0.5, label="", ls=:dot)
18 plot!(x → pdf(normal_U, x), 15-0.5, k;
19       fillrange=0, c=:red, fc=:red, fa=0.5, label="")
20 annotate!(k_U+2, 0.01, text("$(round(100ccdf_U; digits=1))%", :red, :left, 10))
21 annotate!(k-2, 0.01, text("$(round(100cdf_U; digits=1))%", :red, :right, 10))
22
23 vline!([k]; label="k=$k", c=:black, ls=:dot)
24 vline!([n*p_L]; label="n p_L = $(round(n*p_L; digits=2))", c=:blue, ls=:dash)
25 vline!([n*p_U]; label="n p_U = $(round(n*p_U; digits=2))", c=:red, ls=:dashdot)
26 plot!([n*p_L, n*p_U], fill(-0.003, 2); label="", c=:red, lw=5)
27 title!("n = $n, [n p_L, n p_U] = [$(round(n*p_L; digits=2)), $(round(n*p_U; digits=2))]")
28 plot!(); ylim=(-0.007, 0.105));
```

P値達の説明用

In [36]:

```
1 function illustrate_binomial_pvalues()
2     n = 20
3     p = 0.25
4     k = 9
5
6      $\hat{p} = k/n$ 
7      $\hat{\sigma}^2 = n*\hat{p}*(1-\hat{p})$ 
8
9     null = Binomial(n, p)
10     $\mu, \sigma^2 = \text{mean}(null), \text{var}(null)$ 
11    nullname = distname(null)
12
13    f(x) = mypdf(null, x)
14    xlim = (-2, n+1)
15    xlimk = (k-0.5, n+0.5)
16
17    pval = @sprintf "%.03f" pvalue_clopper_pearson(n, k, p)
18    cval = @sprintf "%.03f" ccdf(null, k-1)
19    P1 = plot(title="(1) Clopper-Pearson")
20    plot!(f, xlim...; label="", c=1)
21    plot!(f, xlimk...; label="", fillrange=0, c=1, fc=:red, fa=0.5)
22    vline!([ $\mu$ ]; label=" $\mu=\$ \mu$ ", c=1, lw=0.5, ls=:dash)
23    annotate!(k+5, 4f(k), text("P-value = $pval", 10, :red))
24    plot!([k+7, k+7], [2.4f(k), 3.6f(k)]; arrow=true, c=:red, label="")
25    annotate!(k+6, 3f(k), text("2x", 10, :red, :right))
26    annotate!(k+5, 2f(k), text("one-tailed = $cval", 10, :red))
27    plot!([k+7, k+1], [1.5f(k), 1.5f(k+1)]; arrow=true, c=:red, label="")
28    plot!(); xtick=0:n, tickfontsize=7)
29
30    pval = @sprintf "%.03f" pvalue_sterne(n, k, p)
31    j = _search_boundary(_pdf_le, 2mode(null)-k, -1, (null, pdf(null, k)))
32    xlimj = (-0.5, j+0.5)
33    P2 = plot(title="(2) Sterne")
34    plot!(f, xlim...; label="", c=1)
35    plot!(f, xlimk...; label="", fillrange=0, c=1, fc=:red, fa=0.5)
36    plot!(f, xlimj...; label="", fillrange=0, c=1, fc=:red, fa=0.5)
37    vline!([ $\mu$ ]; label=" $\mu=\$ \mu$ ", c=1, lw=0.5, ls=:dash)
38    plot!([k-0.5, j+0.5], [f(k), f(k)]; label="", c=2, ls=:dash)
39    annotate!(k+5, 3f(k), text("P-value = $pval", 10, :red))
40    plot!([k+5, k+1], [2.5f(k), 0.8f(k)]; arrow=true, c=:red, label="")
41    plot!([k+5, j+1], [2.5f(k), 0.8f(j)]; arrow=true, c=:red, label="")
42    plot!(); xtick=0:n, tickfontsize=7)
43
44    pval = @sprintf "%.03f" pvalue_wilson(n, k, p)
45    normal = Normal( $\mu, \sqrt{\sigma^2}$ )
46    g(x) = pdf(normal, x)
47    P3 = plot(title="(3) Wilson")
48    plot!(f, xlim...; label="", c=1)
49    vline!([ $\mu$ ]; label=" $\mu=\$ \mu$ ", c=1, lw=0.5, ls=:dash)
50    plot!(g, xlim...; ls=:dash, label="Normal( $\mu=\$ \mu, \sigma=\sqrt{\sigma^2}$ )", c=2)
51    plot!(g, k, n; label="", fillrange=0, c=2, fc=:red, fa=0.5)
52    plot!(g, -2, 2 $\mu$ -k; label="", fillrange=0, c=2, fc=:red, fa=0.5)
53    annotate!(k+5, 3f(k), text("P-value = $pval", 10, :red))
54    plot!([k+5, k+0.5], [2.5f(k), 0.8g(k)]; arrow=true, c=:red, label="")
55    plot!([k+5, j+0.2], [2.5f(k), 0.7g(j)]; arrow=true, c=:red, label="")
56    plot!(); xtick=0:n, tickfontsize=7)
57
58    pval = @sprintf "%.03f" pvalue_wald(n, k, p)
59    normal = Normal( $\mu, \sqrt{\hat{\sigma}^2}$ )
60    h(x) = pdf(normal, x)
61    P4 = plot(title="(4) Wald")
62    plot!(f, xlim...; label="", c=1)
63    vline!([ $\mu$ ]; label=" $\mu=\$ \mu$ ", c=1, lw=0.5, ls=:dash)
64    plot!(h, xlim...; ls=:dash, label="Normal( $\mu=\$ \mu, \sigma=\sqrt{\hat{\sigma}^2}$ )", c=2)
65    plot!(h, k, n; label="", fillrange=0, c=2, fc=:red, fa=0.5)
66    plot!(h, -2, 2 $\mu$ -k; label="", fillrange=0, c=2, fc=:red, fa=0.5)
67    annotate!(k+5, 3f(k), text("P-value = $pval", 10, :red))
68    plot!([k+5, k+0.5], [2.5f(k), 0.8g(k)]; arrow=true, c=:red, label="")
69    plot!([k+5, j+0.2], [2.5f(k), 0.7g(j)]; arrow=true, c=:red, label="")
70    plot!(); xtick=0:n, tickfontsize=7)
71
72    plot(P1, P2, P3, P4; size=(800, 520), layout=(2, 2))
73    plot!(); plot_title="model: $nullname, data: n=$n, k=$k")
74 end
75
76 #illustrate_binomial_pvalues()
```

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

```
In [37]: 1 # 二項分布モデルの4種のP値関数のプロット
2
3 function plot_binom_pvaluefunctions(; n = 20, k = 6)
4     P1 = plot(p → pvalue_clopper_pearson(n, k, p), 0, 1;
5         label="", title="Clopper-Pearson", c=1)
6     P2 = plot(p → pvalue_sterne(n, k, p), 0, 1;
7         label="", title="Sterne", c=2)
8     P3 = plot(p → pvalue_wilson(n, k, p), 0, 1;
9         label="", title="Wilson", c=3)
10    P4 = plot(p → pvalue_wald(n, k, p), 0, 1;
11        label="", title="Wald", ls=:dash, c=4)
12    plot(P1, P2, P3, P4; size=(800, 520), layout=(2, 2))
13    plot!(); xtick=0:0.1:1, ytick=0:0.1:1)
14    plot!(); xguide="success rate parameter p",
15        yguide="P-value", guidefontsize=10)
16    plot!(); plot_title="data: n = $n, k = $k",
17        plot_titlefontsize=12)
18    plot!(); titlefontsize=12)
19 end
20
21 #plot_binomial_pvaluefunctions(; n = 20, k = 6)
```

Out[37]: plot_binom_pvaluefunctions (generic function with 1 method)

```
In [38]: 1 # 二項分布モデルのP値関数の動画 (データ k を動かしてアニメ化)
2
3 function gif_binomtest(; n = 20, fn = "images/pvaluefunction20.gif", fps = 5, step=1)
4     anim = @animate for k in [0:step:n; n-1:-step:1]
5         plot(p → pvalue_clopper_pearson(n, k, p), 0, 1; label="Clopper-Pearson")
6         plot!(p → pvalue_sterne(n, k, p), 0, 1; label="Sterne")
7         plot!(p → pvalue_wilson(n, k, p), 0, 1; label="Wilson")
8         plot!(p → pvalue_wald(n, k, p), 0, 1; label="Wald", ls=:dash)
9         plot!(); xtick=0:0.1:1, ytick=0:0.1:1)
10        plot!(); xguide="success rate parameter p", yguide="P-value")
11        title!("data: n = $n, k = $k")
12        plot!(); size=(600, 300))
13        2k > n && plot!(); legend=:topleft)
14    end
15    gif(anim, fn; fps)
16 end
```

Out[38]: gif_binomtest (generic function with 1 method)

```
In [39]: 1 # P値関数と信頼区間を同時プロット
2
3 function plot_binom_pvalue_and_confint(
4     pvalfunc, cifunc, n, k;
5     α = 0.05, name = "", size=(500, 300), kwargs...)
6     p_L, p_U = ci = cifunc(n, k; α)
7     plot(p → pvalfunc(n, k, p), 0, 1;
8         label="P-value function", c=1)
9     plot!(ci, fill(α, 2); label="confidence interval", c=:red, lw=2)
10    vline!([p_L]; label="p_L", c=2, ls=:dash)
11    vline!([p_U]; label="p_U", c=3, ls=:dashdot)
12    plot!(); xtick=0:0.1:1, ytick=0:0.05:1, tickfontsize=7)
13    plot!(); xguide="success rate parameter p", yguide="P-value")
14    title!("$name)n=$n, k=$k, α=$α")
15    plot!(); size, kwargs...)
16 end
17
18 #plot_binom_pvalue_and_confint(
19 #     pvalue_clopper_pearson, confint_clopper_pearson, 20, 6;
20 #     name = "Clopper-Pearson: ")
```

1 比率の検定と信頼区間

以下のようなことを行いたい.

(1) 当たりが未知の確率で出るルーレットを n 回まわしたときに, k 回当たりが出たというデータから, ルーレットで当たりが出る確率を推定する.

(2) 壺の中に赤い玉と白い玉が大量に入っているとき、そこから n 個を無作為抽出したときに k 個が赤い玉であったというデータから、壺の赤い玉の割合を推定する。

このような推定を以下では **比率の推定** と呼ぶことにする。

ルーレットでの当たりの確率や壺の中の赤い玉の割合を総称して **比率** (proportion) と呼んでいる。

目標は比率の信頼区間の構成である。

そのためには、検定と信頼区間の表裏一体性より、P値を適切に定義すればよい。

以下では、ルーレットで当たりが出たり、壺から赤い玉が出ることを **成功** と呼ぶことにする。

注意: 現実世界の統計分析では「成功」は「死亡」を意味する場合もあって、良いことばかりとは限らない。

1.1 P値と信頼区間の定義の復習

比率の二項分布モデルでの取り扱いを例にP値と信頼区間の定義を復習しよう。

データ: n 回中 k 回成功

統計モデル: 統計モデルとして二項分布を使用する:

$$P(k|n, p) = \binom{n}{k} p^k (1-p)^{n-k} \quad (k = 0, 1, 2, \dots, n)$$

二項分布の累積分布関数を次のように書く:

$$\text{cdf}(\text{Binomial}(n, p), k) = \sum_{i \leq k} P(j|n, p).$$

他の分布の累積分布関数も同様に書くことにする。

検定される仮説: $p = p_0$ (p_0 は具体的な数値)

P値の大雑把な定義: 仮説 $p = p_0$ 下の統計モデル内でデータの数値「 n 回中 k 回成功」以上に極端な値が生じる確率またはその近似値。

「データの数値以上に極端な」の正確な定義や近似の仕方については別に与えることになる。

P値の使い方: データの数値「 n 回中 k 回成功」に関する仮説 $p = p_0$ のP値は、仮説 $p = p_0$ 下の **統計モデルとデータの数値の整合性** の指標として使われる。

以下ではP値を次のように書く:

$$\text{pvalue}(k|n, p = p_0) = (\text{データ「}n\text{ 回中 }k\text{ 回成功」に関する仮説 }p = p_0\text{ のP値}).$$

有意水準: $0 \leq \alpha \leq 1$ (5% や 1% など小さな値にする)。

検定: P値が有意水準未満ならば仮説 $p = p_0$ は **棄却** されたという。

ただし、これは閾値 α に基いて、単に仮説 $p = p_0$ 下の統計モデルとデータの数値の整合性が低過ぎると判断することを超えた意味を持たないことには注意しなければいけない。

データの取得法に問題があった可能性や仮説 $p = p_0$ も含めた統計モデルの前提のどれかが妥当でない可能性を疑うことになる。

仮説 $p = p_0$ が棄却されない場合にはその妥当性に関する判断を保留する。

信頼度: $1 - \alpha$ 。

信頼区間: 検定で棄却されない成功確率パラメータ $p = p_0$ の全体の集合。

以下では信頼区間(confidence interval)を次のように書く:

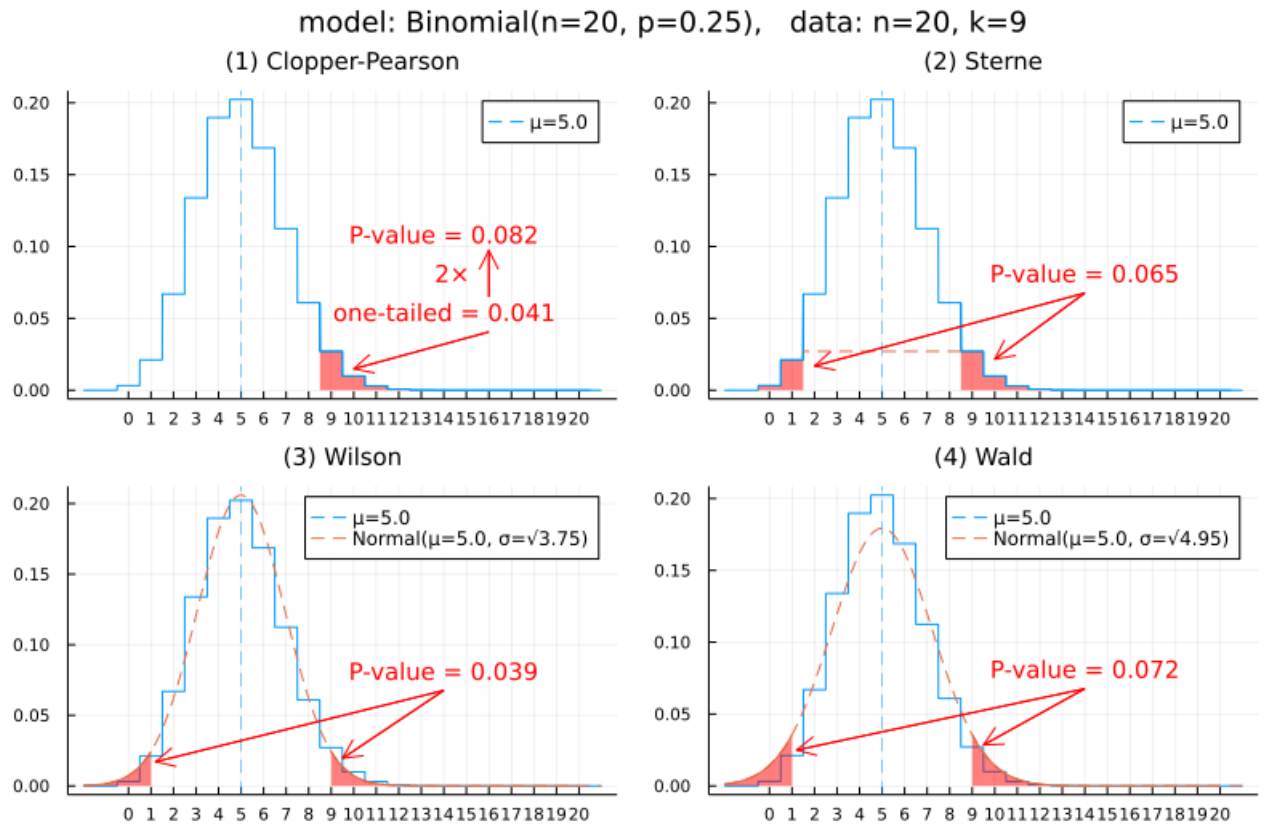
$$\text{confint}(k|n, \alpha) = \{p_0 \in [0, 1] \mid \text{pvalue}(k|n, p = p_0) \geq \alpha\}.$$

信頼区間の使い方: 信頼区間に含まれるパラメータ値 $p = p_0$ についてはその妥当性についての判断は保留する。信頼区間に含まれないパラメータ値 $p = p_0$ については、仮説 $p = p_0$ が検定によって棄却されたと考える。(注意: 実際には統計モデルのすべての前提が疑われることになる。)

1.2 二項分布モデルのP値達の定義を1枚の図で要約

```
In [40]: 1 illustrate_binomial_pvalues()
```

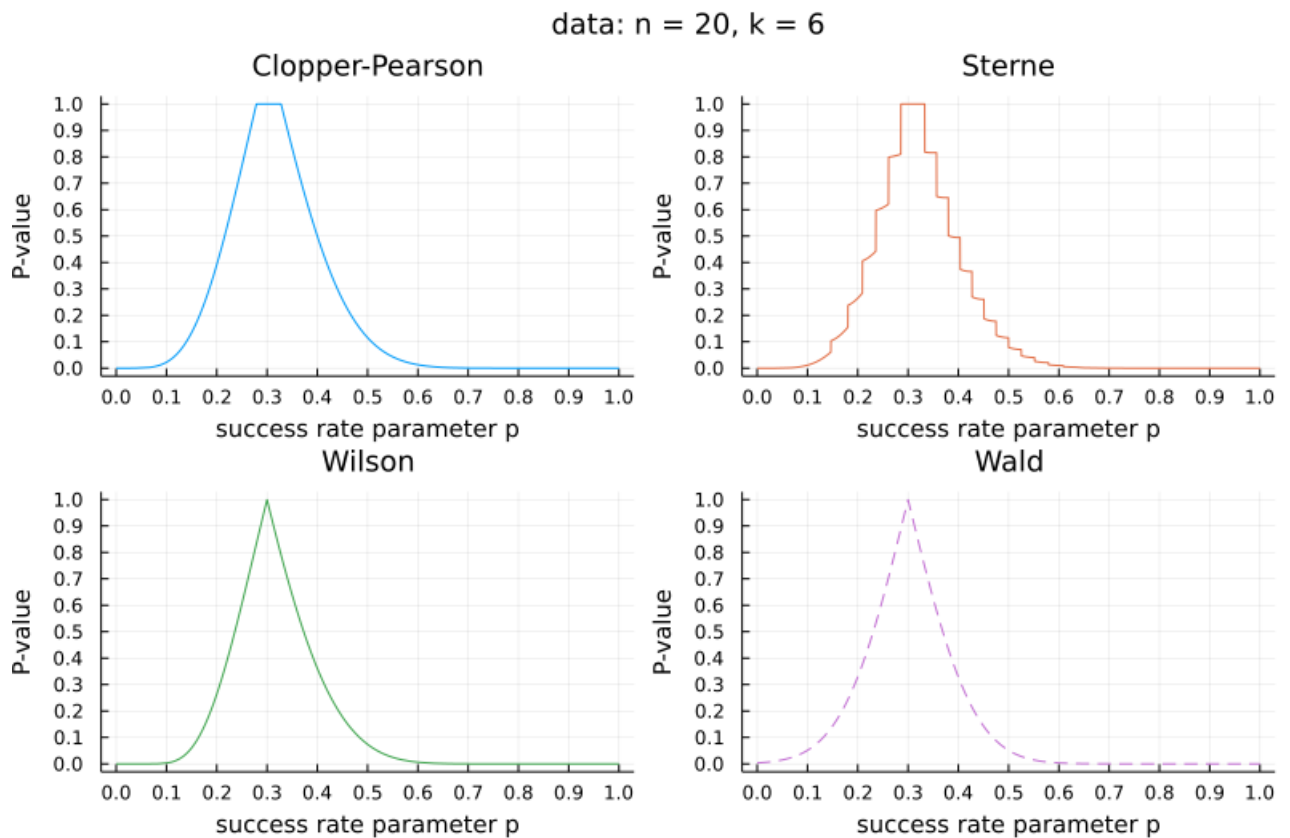
Out[40]:



1.3 4種のP値関数のプロット

```
In [41]: 1 plot_binom_pvaluefunctions()
```

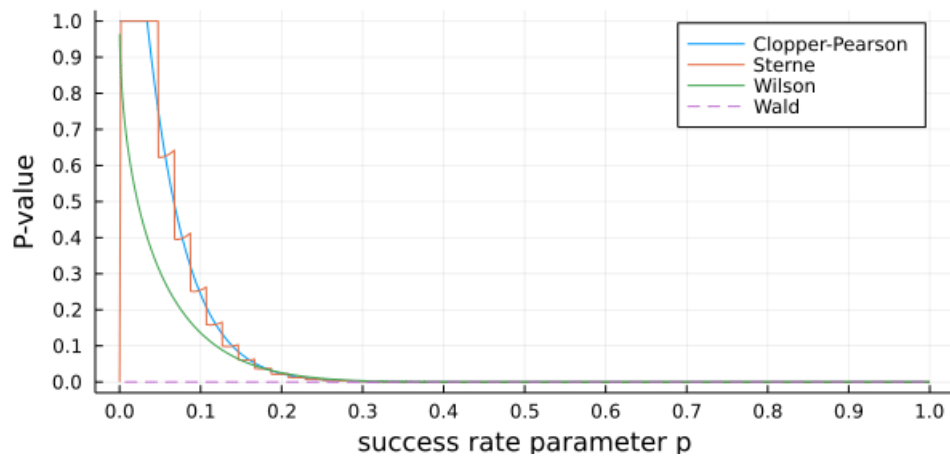
Out[41]:



```
In [42]: 1 gif_binomtest()
```

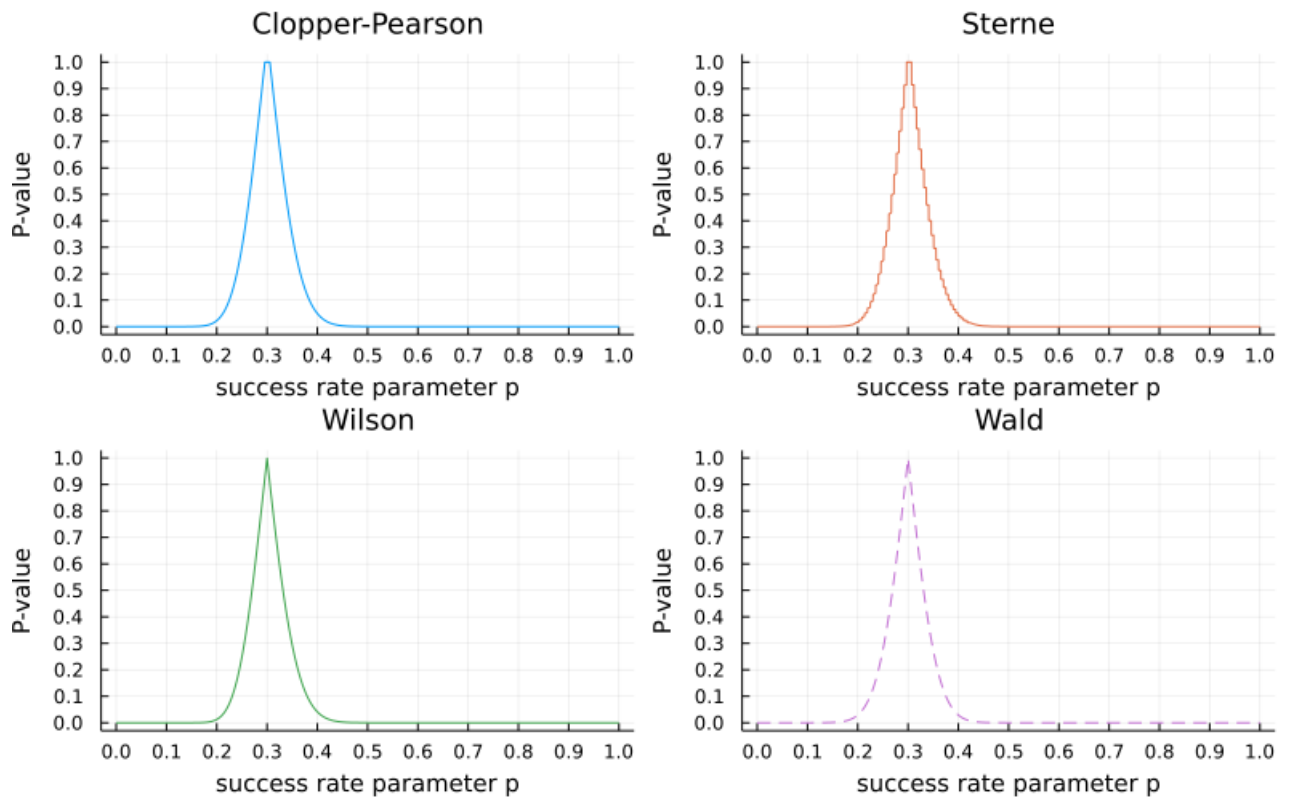
```
Info: Saved animation to  
fn = D:\OneDrive\work\Statistics\2022\images\pvaluefunction20.gif  
@ Plots D:\.julia\packages\Plots\mnkn4\src\animation.jl:114
```

Out[42]: data: n = 20, k = 0



```
In [43]: 1 plot_binom_pvaluefunctions(n = 100, k = 30)
```

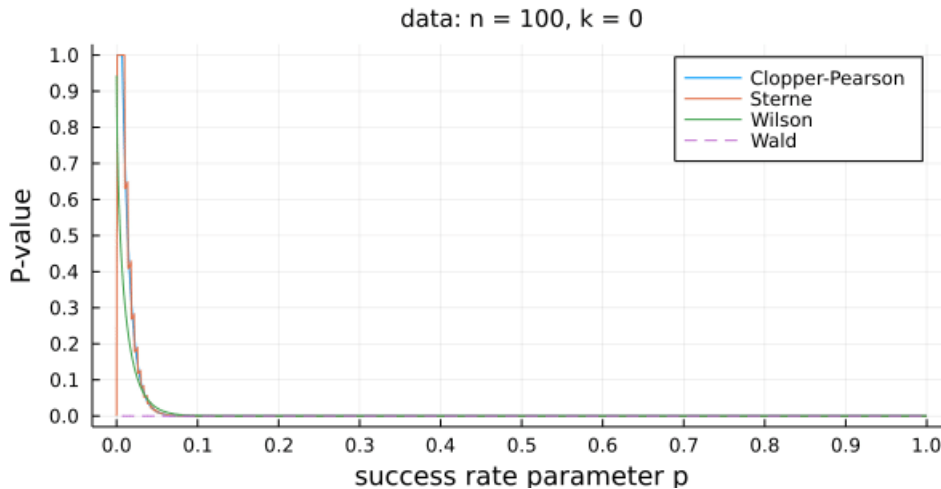
Out[43]: data: n = 100, k = 30




```
In [44]: 1 gif_binomtest(; n = 100, fn = "images/pvaluefunction100.gif", fps = 20)
```

```
Info: Saved animation to  
fn = D:\OneDrive\work\Statistics\2022\images\pvaluefunction100.gif  
@ Plots D:\.julia\packages\Plots\mnkn4\src\animation.jl:114
```

Out[44]:



PDFファイルではこの動画を見ることはできない. 作成した動画は

- <https://github.com/genkuroki/Statistics/blob/master/2022/images/pvaluefunction20.gif>
(<https://github.com/genkuroki/Statistics/blob/master/2022/images/pvaluefunction20.gif>)
- <https://github.com/genkuroki/Statistics/blob/master/2022/images/pvaluefunction100.gif>
(<https://github.com/genkuroki/Statistics/blob/master/2022/images/pvaluefunction100.gif>)

でも見るができる.

2 Clopper-Pearson

2.1 Clopper-Pearsonの信頼区間を与えるP値関数

データ「 n 回中 k 回成功」について, 「データの数値以上に極端な」の意味を「 k 以上の」または「 k 以下の」の確率の小さい方とした場合. ただし, P値はその確率の2倍として定義する(1 を超えないように 1 より大きくなったら 1 に切り詰める):

$$\text{pvalue}_{\text{CP}}(x|n, p = p_0) = \min \left(\begin{array}{c} 1 \\ 2 \text{cdf}(\text{Binomial}(n, p_0), k) \\ 2(1 - \text{cdf}(\text{Binomial}(n, p_0), k - 1)) \end{array} \right)$$

$\text{cdf}(\text{Binomial}(n, p_0), k)$ は二項分布の累積分布関数なので, 二項分布 $\text{Binomial}(n, p_0)$ 内で k 以下になる確率であり, $1 - \text{cdf}(\text{Binomial}(n, p_0), k - 1)$ は k 以上になる確率である.

2.2 Clopper-Pearsonの信頼区間

p_L, p_U は次の条件で定義する:

$$\begin{aligned} 1 - \text{cdf}(\text{Binomial}(n, p_L), k - 1) &= \alpha/2, \\ \text{cdf}(\text{Binomial}(n, p_U), k) &= \alpha/2. \end{aligned}$$

前節でのP値の定義は次のClopper-Pearsonの信頼区間を与える:

$$\text{confint}_{\text{CP}}(k|n, \alpha) = [p_L, p_U].$$

実際の計算の方法: 二項分布とベータ分布の関係

$$1 - \text{cdf}(\text{Binomial}(n, p), k - 1) = \text{cdf}(\text{Beta}(k, n - k + 1), p),$$

$$\text{cdf}(\text{Binomial}(n, p), k) = 1 - \text{cdf}(\text{Beta}(k + 1, n - k), p)$$

と分位点関数 `quantile` が累積分布関数 `cdf` の逆関数であることより,

$$p_L = \text{quantile}(\text{Beta}(k, n - k + 1), \alpha/2),$$

$$p_U = \text{quantile}(\text{Beta}(k + 1, n - k), 1 - \alpha/2).$$

によって, p_L, p_U を計算可能である.

2.3 Clopper-Pearsonの信頼区間の図による把握

以下の図をよく見れば, $\alpha = 5\%$ の場合の $p_L = p_{\text{L}}$ と $p_U = p_{\text{U}}$ を直観的に把握できるだろう.

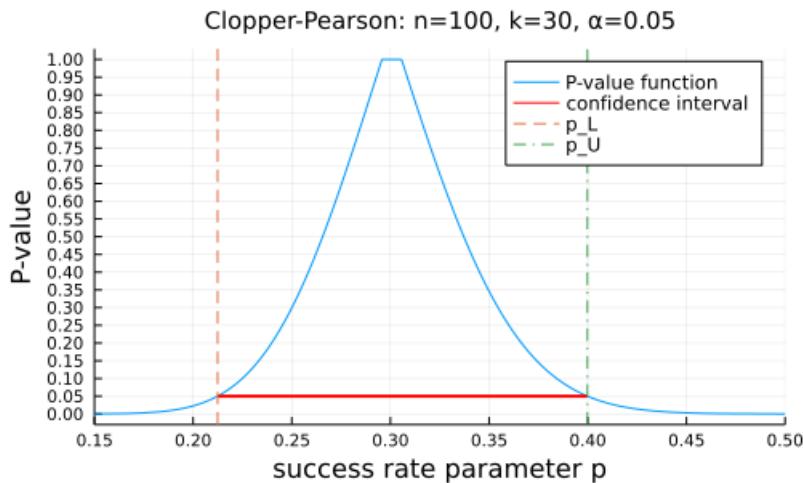
まず, データの数値が与えられたとき, モデルのパラメータ値をP値に対応させる **P値関数** (P-value function)の立場では, **信頼区間** (confidence interval)の解釈は易しい.

信頼区間はP値関数が有意水準の値を取る2つのパラメータに挟まれた区間になる.

しかし, P値関数全体のグラフを描けば, 信頼区間を計算しなくても, データとの整合性が低過ぎないパラメータ値の範囲は明らかだろう.

```
In [45]: 1 plot_binom_pvalue_and_confint(
2         pvalue_clopper_pearson, confint_clopper_pearson, 100, 30;
3         name = "Clopper-Pearson: ", xlim=(0.15, 0.5), xtick=0:0.05:1)
```

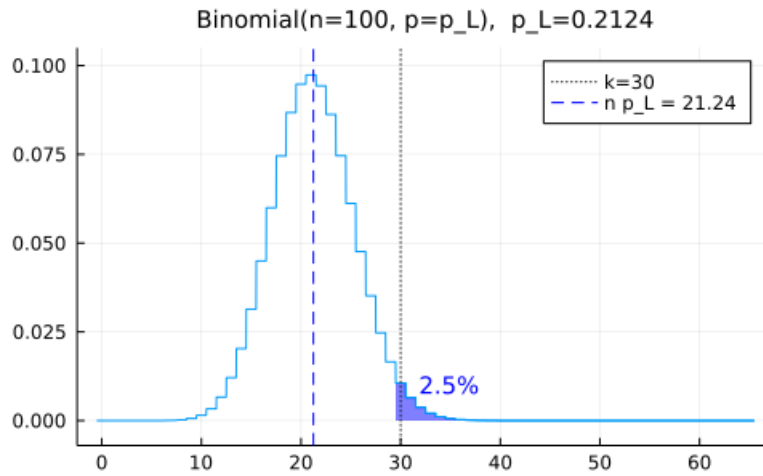
Out[45]:



$1 - \alpha = 95\%$ 信頼区間の下限を求めるためには, 二項分布モデルの成功確率パラメータ p を小さくして行って, モデル内でデータの数値 k 以上の値が生成される確率の2倍 (= Clopper-Pearsonの信頼区間を与えるP値)がちょうど $\alpha = 5\%$ になる p の値を見つければよい. そのとき, モデル内でデータの数値 k 以上の値が生成される確率は $\alpha/2 = 2.5\%$ になっている. そのような p を $p_L = p_{\text{L}}$ とする.

```
In [46]: 1 var"Clopper-Pearson: 下側では  $p = p_L$  がぎりぎり"
```

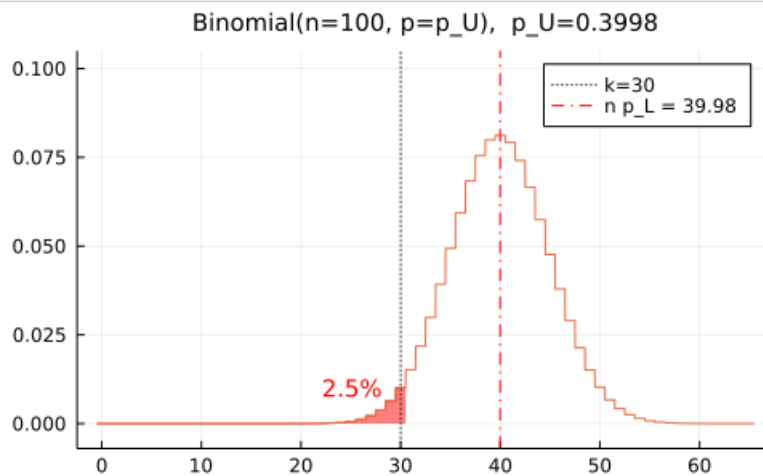
Out[46]:



$1 - \alpha = 95\%$ 信頼区間の上限を求めるためには、二項分布モデルの成功確率パラメータ p を大きくして行って、モデル内でデータの数値 k 以下の値が生成される確率の2倍(=Clopper-Pearsonの信頼区間を与えるP値)がちょうど $\alpha = 5\%$ になる p の値を見つければよい。そのとき、モデル内でデータの数値 k 以下の値が生成される確率は $\alpha/2 = 2.5\%$ になっている。そのような p を $p_U = p_U$ とする。

```
In [47]: 1 var"Clopper-Pearson: 上側では  $p = p_U$  がぎりぎり"
```

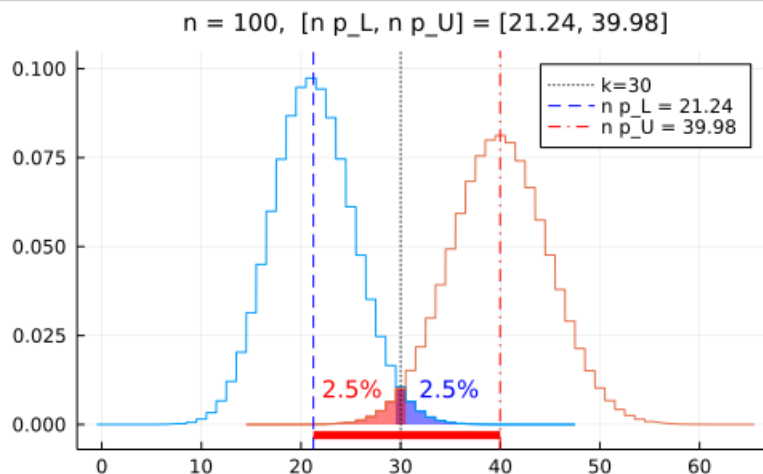
Out[47]:



以上をまとめると以下のような図を描ける。

```
In [48]: 1 var"Clopper-Pearson:  $n \times 95\%$  信頼区間"
```

Out[48]:



この場合のClopper-Pearsonの信頼区間は $[0.2124, 0.3998]$ になっている。

以上の図は「例：ベータ関数と二項分布の関係とその応用」のノート

([https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/07-](https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/07-1%20Relationship%20between%20beta%20and%20binomial%20distributions.ipynb)

[1%20Relationship%20between%20beta%20and%20binomial%20distributions.ipynb](https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/07-1%20Relationship%20between%20beta%20and%20binomial%20distributions.ipynb))の「Clopper-Pearsonの信頼区間の定義」の

節で使ったのと同じものである。

3 Sterne

3.1 Sterneの信頼区間を与えるP値関数

「データの数値以上に極端な」の意味を「生成される確率がデータの数値以下の」とした場合:

$$\text{pvalue}_{\text{Stern}}(k|n, p = p_0) = \sum_{j \text{ satisfying } P(j|n, p_0) \leq P(k|n, p_0)} P(j|n, p_0).$$

これは、モデル内での生成確率 $P(j|n, p_0)$ がデータの数値と同じ値の生成確率 $P(k|n, p_0)$ 以下になる j について確率 $P(j|n, p_0)$ を足し上げたものである。

注意: Clopper-Pearsonの信頼区間を与えるP値もSterneの信頼区間を与えるP値も第一種の過誤の確率が常に有意水準以下になるという意味で保守的(conservative)なP値の定義になっている。そして、Sterneの信頼区間を与えるP値の方が小さな値になることが多いということも知られている。この基準ではSterneの信頼区間を与えるP値の方がClopper-Pearsonの信頼区間を与えるP値よりも優れていると言える。しかし、Sterneの信頼区間を与えるP値の効率的実装はClopper-Pearsonの信頼区間を与えるP値の実装よりもずっと複雑になってしまう。

3.2 Sterneの信頼区間

上で定義したP値の定義はSterneの信頼区間([Sterne \(1954\) \(https://www.jstor.org/stable/2333026\)](https://www.jstor.org/stable/2333026))を与える:

$$\text{confint}_{\text{Stern}}(k|n, \alpha) = \{p_0 \in [0, 1] \mid \text{pvalue}_{\text{Stern}}(k|n, p = p_0) \geq \alpha\}.$$

Sterneの信頼区間は被覆確率(coverage probability)や長さの期待値(expected length)の基準から、Clopper-Pearsonの信頼区間よりも優れていることが知られている。

しかし、効率的な計算の実装はずっと複雑になり、Sterneの信頼区間の計算のためにはコンピュータの取り扱いに習熟する必要がある。このノートでは読者にそこまで要求するつもりはない。しかし、プログラミングに自信がある人達はSterneの信頼区間の実装を自力で試してみると大変良い勉強になると思われる。

注意: このタイプの信頼区間は稀に区間にならない場合がある。そのような稀な場合にも区間にするにこだわるならば、P値が α 以上になる $p = p_0$ 全体の集合を含む最小の区間を改めて信頼区間の定義とするというような処置が必要になる。その場合には以下ようになる:

$$\text{confint}_{\text{Stern}}(k|n, \alpha) = [p_{\text{inf}}, p_{\text{sup}}].$$

ここで、

$$p_{\text{inf}} = \inf\{p_0 \in [0, 1] \mid \text{pvalue}_{\text{Stern}}(k|n, p = p_0) \geq \alpha\},$$
$$p_{\text{sup}} = \sup\{p_0 \in [0, 1] \mid \text{pvalue}_{\text{Stern}}(k|n, p = p_0) \geq \alpha\}.$$

注意: Sterneの信頼区間とClopper-Pearsonの信頼区間などの比較については例えば以下の報告がある:

- 原茂恵美子, 武藤彬正, 宮島育哉, 榊原伊織, SASによる二項比率における正確な信頼区間の比較, 2014. [[pdf \(https://www.sas.com/content/dam/SAS/ja_jp/doc/event/sas-user-groups/usergroups14-d-05.pdf\)](https://www.sas.com/content/dam/SAS/ja_jp/doc/event/sas-user-groups/usergroups14-d-05.pdf)]

3.3 Sterneの信頼区間の図による把握

以下の図をよく見れば、 $\alpha = 5\%$ の場合の $p_L = p_{-L}$ と $p_U = p_{+U}$ を直観的に把握できるだろう。

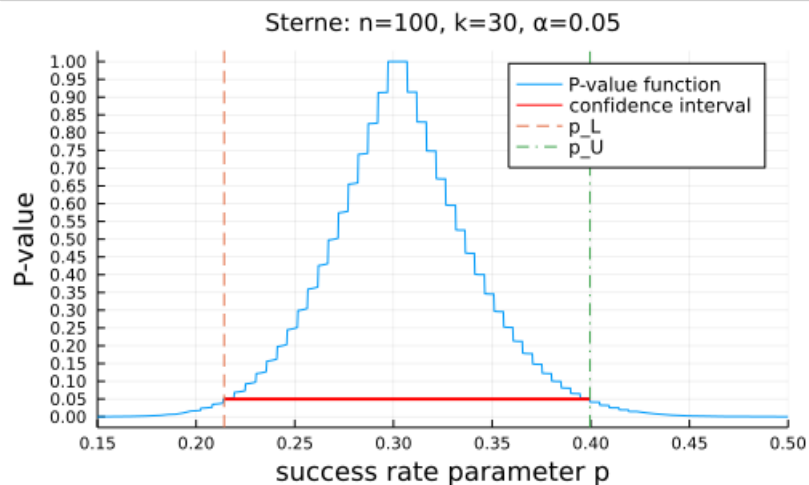
まず、データの数値が与えられたとき、モデルのパラメータ値をP値に対応させる **P値関数** (P-value function)の立場では、**信頼区間** (confidence interval)の解釈は易しい。

信頼区間はP値関数が有意水準の値を取る2つのパラメータに挟まれた区間になる。

しかし、P値関数全体のグラフを描けば、信頼区間を計算しなくても、データとの整合性が低過ぎないパラメータ値の範囲は明らかだろう。

```
In [49]: 1 plot_binom_pvalue_and_confint(
2         pvalue_sterne, confint_sterne, 100, 30;
3         name = "Sterne: ", xlim=(0.15, 0.5), xtick=0:0.05:1)
```

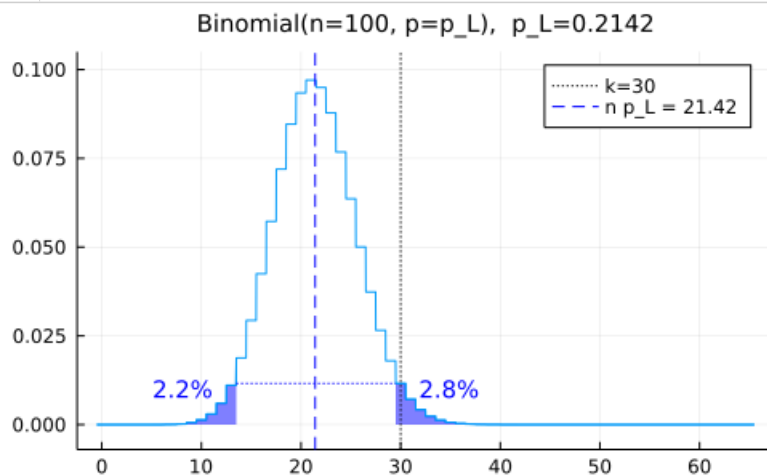
Out[49]:



$1 - \alpha = 95\%$ 信頼区間の下限を求めるためには、二項分布モデルの成功確率パラメータ p を小さくして行って、モデル内でデータの数値 k 以上の値が生成される確率とその反対側の確率の和 (= Sterneの信頼区間を与えるP値)が $\alpha = 5\%$ 以上になるような p の値の下限を見つければよい. そのような p を $p_L = p_{\text{L}}$ とする.

```
In [50]: 1 var"Sterne: 下側では p = p_L がぎりぎり"
```

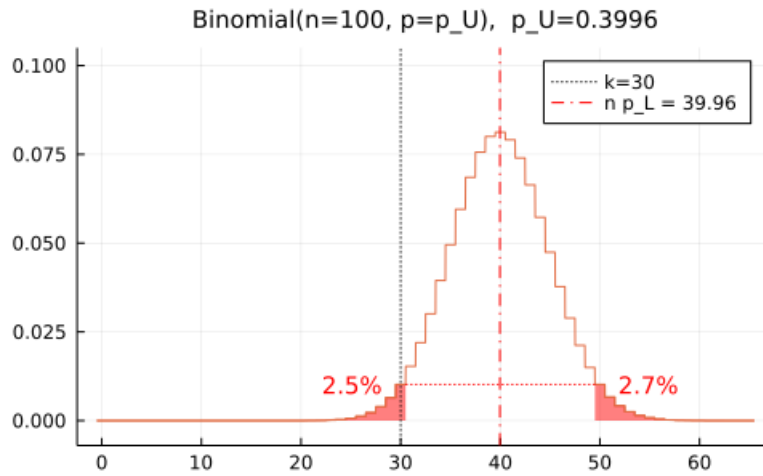
Out[50]:



$1 - \alpha = 95\%$ 信頼区間の上限を求めるためには、二項分布モデルの成功確率パラメータ p を大きくして行って、モデル内でデータの数値 k 以下の値が生成される確率とその反対側の確率の和 (= Sterneの信頼区間を与えるP値)が $\alpha = 5\%$ 以上になるような p の値の上限を見つければよい. そのような p を $p_U = p_{\text{U}}$ とする.

```
In [51]: 1 var"Sterne: 上側では p = p_U がぎりぎり"
```

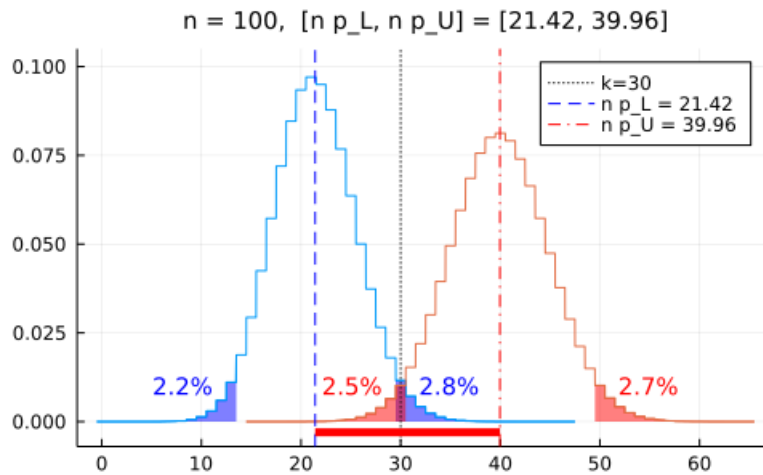
Out[51]:



以上をまとめると以下のような図を描ける.

```
In [52]: 1 var"Sterne: n × 95%信頼区間"
```

Out[52]:



この場合のSterneの信頼区間は $[0.2142, 0.3996]$ になっている.

これは、同じ場合のClopper-Pearsonの信頼区間の $[0.2124, 0.3998]$ よりもほんの少し狭いが、ほぼ同じであると言ってよいだろう.

3.4 おまけ: R言語のbinom.testについて

R言語の `binom.test` がデフォルトで表示するP値はSterneの信頼区間を与えるP値であるが、それと同時に表示される信頼区間はClopper-Pearsonの信頼区間の方になってしまっている. そのせいで、P値が有意水準未満なのに、帰無仮説を与える成功確率パラメータ値が信頼区間に含まれてしまう場合がある.

```
In [53]: 1 R"binom.test(10, 20, p = 0.72, conf.level = 0.95)"
```

```
Out[53]: RObject{VecSxp}
```

```
Exact binomial test

data: 10 and 20
number of successes = 10, number of trials = 20, p-value = 0.04277
alternative hypothesis: true probability of success is not equal to 0.72
95 percent confidence interval:
 0.2719578 0.7280422
sample estimates:
probability of success
              0.5
```

上のRの binom.test の結果と以下を比較してみよ.

```
In [54]: 1  $\alpha$  = 0.05
2 n, k = 20, 10
3 null = Binomial(n, 0.72)
```

```
Out[54]: Binomial{Float64}(n=20, p=0.72)
```

```
In [55]: 1 # Sterneの信頼区間を与えるP値を定義通りに計算(計算効率はよくない)
2 p_value_sterne =
3     sum(pdf(null, i) for i in 0:n if pdf(null, i)  $\leq$  pdf(null, k))
4 @show p_value_sterne;
```

```
p_value_sterne = 0.04276902774675169
```

```
In [56]: 1 # p を pvalue_sterne -  $\alpha$  に対応させる函数が
2 # 0 になる 0 から 1 のあいだの p 達を求めれば,
3 # Sterneの信頼区間を計算できる.
4 conf_int_sterne = find_zeros(0, 1) do p
5     null = Binomial(n, p)
6     pvalue_sterne =
7         sum(pdf(null, i) for i in 0:n if pdf(null, i)  $\leq$  pdf(null, k))
8     pvalue_sterne -  $\alpha$ 
9 end
10 @show conf_int_sterne;
```

```
conf_int_sterne = [0.2927095227968867, 0.7072904772031132]
```

[f\(a, b\) do ... end](https://docs.julialang.org/en/v1/manual/functions/#Do-Block-Syntax-for-Function-Arguments) 構文の解説 (<https://docs.julialang.org/en/v1/manual/functions/#Do-Block-Syntax-for-Function-Arguments>)

```
In [57]: 1 # Clopper-Pearsonの信頼区間を与えるP値を定義通りに計算
2 p_value_clopper_pearson = min(1, 2cdf(null, k), 2(1-cdf(null, k-1)))
3 @show p_value_clopper_pearson;
```

```
p_value_clopper_pearson = 0.06093072477502701
```

```
In [58]: 1 # Clopper-Pearsonの信頼区間をベータ分布を使って計算
2 conf_int_clopper_pearson =
3     [quantile(Beta(k, n-k+1),  $\alpha$ /2), quantile(Beta(k+1, n-k), 1- $\alpha$ /2)]
4 @show conf_int_clopper_pearson;
```

```
conf_int_clopper_pearson = [0.27195784956079183, 0.7280421504392082]
```

有意水準が $\alpha = 5\%$ で、「20 回中 10 回成功」というデータが得られたときに、仮説「成功確率は $p = 0.72$ である」のP値と成功確率 p の信頼区間は以下の表ようになる:

計算法	仮説 $p = 0.72$ のP値	信頼区間
Sterne	$4.277\% < \alpha = 5\%$	$0.72 \notin [0.2927095, 0.7072905]$
Clopper-Pearson	$6.093\% > \alpha = 5\%$	$0.72 \in [0.2719578, 0.7280422]$

このようにP値と信頼区間の表裏一体性に忠実であれば、P値が有意水準未満であることと、信頼区間に帰無仮説を与えるパラメータ値が含まれないことが同値になる。しかし、R言語の binom.test は信頼区間の表裏一体正を無視しているので、整合性がない結果が稀に表示されることがあるので注意が必要である。

4 Wilson

4.1 Wilsonの信頼区間を与えるP値関数

データ「 n 回中 k 回成功」について、「データの数値以上に極端な」の意味を二項分布の正規分布近似のもとで「 k 以上または k 以下の」とした場合:

$$\text{pvalue}_{\text{Wilson}}(k|n, p = p_0) = 2 \left(1 - \text{cdf} \left(\text{Normal}(0, 1), \frac{|k - np_0|}{\sqrt{np_0(1 - p_0)}} \right) \right).$$

右辺は標準正規分布において絶対値が $|k - np_0|/\sqrt{np_0(1 - p_0)}$ 以上になる確率である。

4.2 Wilsonの信頼区間

以下で使う記号を準備しよう. $\hat{p}, \widehat{SE}, SE$ を以下のように定義する:

$$\hat{p} = \frac{k}{n}, \quad \widehat{SE} = \sqrt{\frac{\hat{p}(1 - \hat{p})}{n}}, \quad SE = \sqrt{\frac{p(1 - p)}{n}}.$$

ハットが付いている記号はデータの数値から計算した推定値であり, SE は標準誤差(standard error)の略である(SE は k が二項分布 $\text{Binomial}(n, p)$ に従ってランダムに生成されているときの \hat{p} の標準偏差になる).

次が成立していることにも注意せよ(左辺の分子分布を n で割る):

$$\frac{k - np}{\sqrt{np_0(1 - p)}} = \frac{\hat{p} - p}{SE}.$$

k が二項分布 $\text{Binomial}(n, p)$ に従ってランダムに生成されているとき, これは近似的に標準正規分布に従うのであった.

この場合の信頼区間は以下のように記述される.

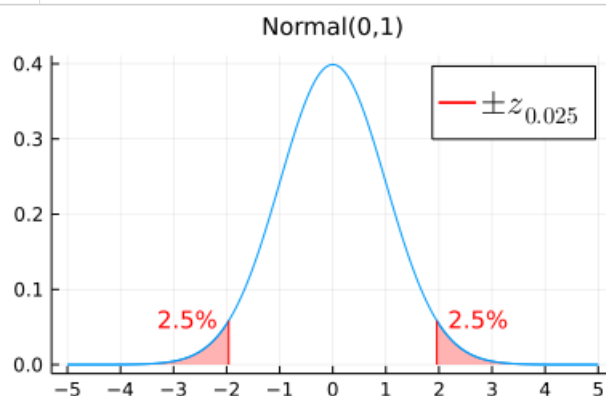
まず, $z_{\alpha/2}$ を次のように定める:

$$z_{\alpha/2} = \text{quantile}(\text{Normal}(0, 1), 1 - \alpha/2).$$

これは標準正規分布において $z_{\alpha/2}$ 以上になる確率が $\alpha/2$ になるという意味である.

In [59]: 1 var"z_{0.025} の定義"

Out[59]:



以下, 記号の簡単のため次のようにおく:

$$z = z_{\alpha/2}.$$

このとき, 標準正規分布において絶対値が $(\hat{p} - p)/SE$ の絶対値以上の値が生成される確率が α に等しいことと

$$\frac{(\hat{p} - p)^2}{SE^2} = z_{\alpha/2}^2 = z^2$$

となることは同値である. $SE^2 = p(1 - p)/n$ なので, これは p に関する次の二次方程式と同値である:

$$\left(1 + \frac{z^2}{n}\right)p^2 - 2\left(\hat{p} + \frac{z^2}{2n}\right)p + \hat{p}^2 = 0.$$

そこで, a, b, c を

$$a = 1 + \frac{z^2}{n}, \quad b = \hat{p} + \frac{z^2}{2n}, \quad c = \hat{p}^2$$

とおく. このとき, 上の二次方程式の2つの解を $p_- \leq p_+$ と書く:

$$p_{\pm} = \frac{b \pm \sqrt{b^2 - ac}}{a} = \frac{1}{1 + z^2/n} \left(\hat{p} + \frac{z^2}{2n} \pm z \sqrt{\widehat{SE}^2 + \frac{z^2}{4n^2}} \right).$$

このとき,

$$\text{confint}_{\text{Wilson}}(k|n, \alpha) := \{p \in [0, 1] \mid \text{pvalue}_{\text{Wilson}}(k|n, p) \geq \alpha\} = [p_-, p_+].$$

これを **Wilsonの信頼区間** と呼ぶ.

注意: Wilsonの信頼区間を与えるP値はシンプルに定義されたが, 対応する信頼区間の計算には二次方程式を解く必要がある. 次の節で説明するWaldの信頼区間はこの部分を簡略化したものとみなされる.

注意: 上で省略した計算の細部は自分で埋めよ. Wilsonの信頼区間の導出については [「例: ベータ関数と二項分布の関係とその応用」のノート](https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/07-1%20Relationship%20between%20beta%20and%20binomial%20distributions.ipynb) (<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/07-1%20Relationship%20between%20beta%20and%20binomial%20distributions.ipynb>) の「Wilsonの信頼区間の基礎」の節も参照せよ.

4.3 Wilsonの信頼区間の図による把握

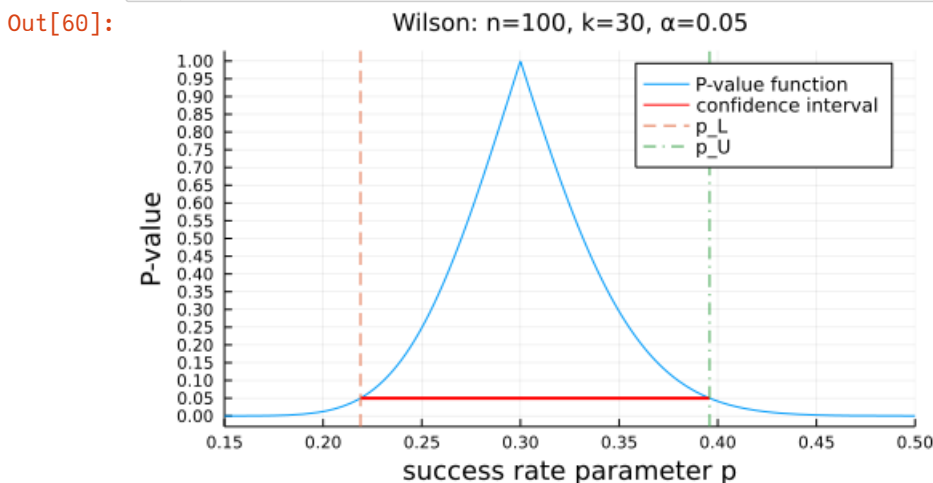
以下の図をよく見れば, $\alpha = 5\%$ の場合の $p_L = p_-L$ と $p_U = p_+U$ を直観的に把握できるだろう.

まず, データの数値が与えられたとき, モデルのパラメータ値をP値に対応させる **P値関数** (P-value function) の立場では, **信頼区間** (confidence interval) の解釈は易しい.

信頼区間はP値関数が有意水準の値を取る2つのパラメータに挟まれた区間になる.

しかし, P値関数全体のグラフを描けば, 信頼区間を計算しなくても, データとの整合性が低過ぎないパラメータ値の範囲は明らかだろう.

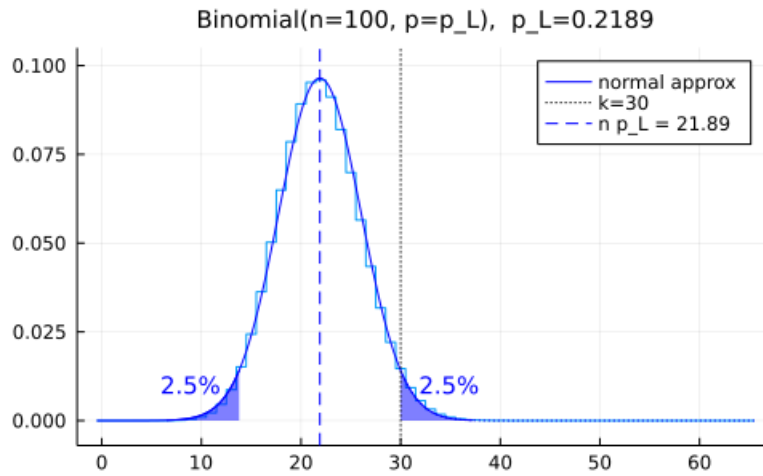
```
In [60]: 1 plot_binom_pvalue_and_confint(
2         pvalue_wilson, confint_wilson, 100, 30;
3         name = "Wilson: ", xlim=(0.15, 0.5), xtick=0:0.05:1)
```



$1 - \alpha = 95\%$ 信頼区間の下限を求めるためには, 二項分布モデルの成功確率パラメータ p を小さくして行って, その二項分布を近似する正規分布内でデータの数値 k 以上の値が生成される確率とその反対側の確率の和 (= Wilsonの信頼区間を与えるP値) がちょうど $\alpha = 5\%$ になるような p を見つければよい. そのような p を $p_L = p_-L$ とする.

```
In [61]: 1 var"Wilson: 下側では p = p_L がぎりぎり"
```

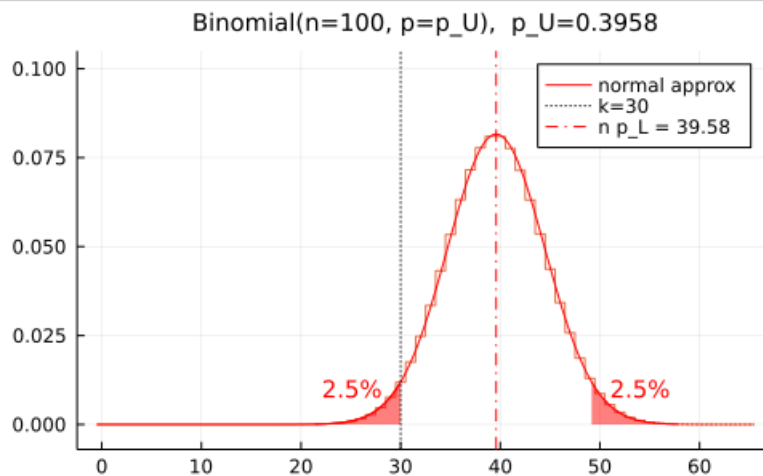
Out[61]:



$1 - \alpha = 95\%$ 信頼区間の上限を求めるためには、二項分布モデルの成功確率パラメータ p を大きくして行って、その二項分布を近似する正規分布内でデータの数値 k 以下の値が生成される確率とその反対側の確率の和 (= Wilsonの信頼区間を与えるP値) がちょうど $\alpha = 5\%$ になるような p を見つければよい. そのような p を $p_U = p_U$ とする.

```
In [62]: 1 var"Wilson: 上側では p = p_U がぎりぎり"
```

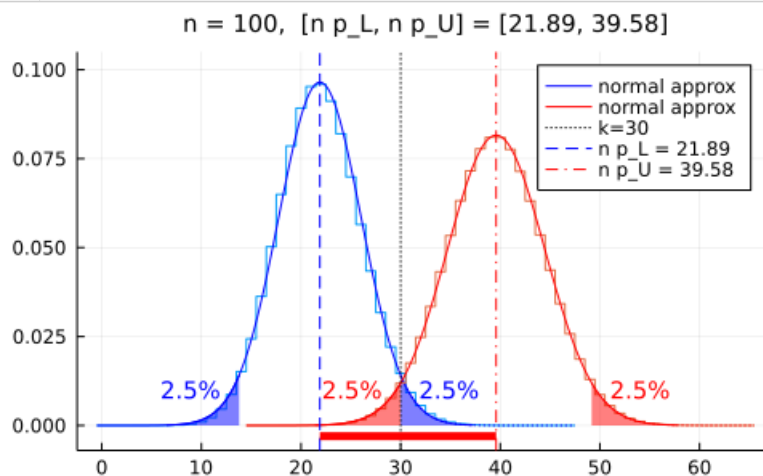
Out[62]:



以上をまとめると以下のような図を描ける.

```
In [63]: 1 var"Wilson: n × 95%信 頼 区 間"
```

Out[63]:



この場合のWilsonの信頼区間は $[0.2189, 0.3958]$ である.

これは、同じ場合のClopper-Pearsonの信頼区間の $[0.2124, 0.3998]$ とSterneの信頼区間は $[0.2142, 0.3996]$ の両方よりも少し狭いが、ほぼ同じであると言ってよいだろう.

4.4 おまけ: R言語のprop.testについて

R言語では `prop.test` 関数でWilsonの信頼区間を与えるP値とWilsonの信頼区間を計算できる. ただし, `correct = FALSE` として, 連続性補正(continuity correction)をオフにしなければ, 以上で説明したWilsonの信頼区間およびそれを与えるP値とはぴったりに一致しない値が表示される.

連続性補正については「[大数の法則と中心極限定理](#)」のノート

(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/05%20Central%20limit%20theorem.ipynb>)の「連続性補正とmid-P補正」の節を参照せよ.

```
In [64]: 1 # これは連続性補正有りの計算結果になっており,  
2 # 上で説明した補正無しのWilsonの信頼区間とそれを与えるP値とは異なる.  
3 R"prop.test(10, 20, p = 0.72, conf.level = 0.95)"
```

```
Out[64]: RObject{VecSxp}  
  
1-sample proportions test with continuity correction  
  
data: 10 out of 20, null probability 0.72  
X-squared = 3.7723, df = 1, p-value = 0.05211  
alternative hypothesis: true p is not equal to 0.72  
95 percent confidence interval:  
 0.2785367 0.7214633  
sample estimates:  
 p  
0.5
```

```
In [65]: 1 R"prop.test(10, 20, p = 0.72, conf.level = 0.95, correct = FALSE)"
```

```
Out[65]: RObject{VecSxp}  
  
1-sample proportions test without continuity correction  
  
data: 10 out of 20, null probability 0.72  
X-squared = 4.8016, df = 1, p-value = 0.02843  
alternative hypothesis: true p is not equal to 0.72  
95 percent confidence interval:  
 0.299298 0.700702  
sample estimates:  
 p  
0.5
```

以上のR言語による計算結果と以下のセルでの計算結果を比較してみよ.

In [66]:

```
1 n, k, p, α = 20, 10, 0.72, 0.05
2 @show n k p α
3 println()
4 @show z = quantile(Normal(), 1-α/2)
5 @show p̂ = k/n
6 @show SE = √(p*(1-p)/n)
7 @show p̂_L, p̂_U = (k - 0.5)/n, (k + 0.5)/n
8
9 p_value_wilson_corrected = 2ccdf(Normal(), (abs(p̂ - p)-0.5/n)/SE)
10 a, b_L, c_L, b_U, c_U = 1+z^2/n, p̂_L+z^2/(2n), p̂_L^2, p̂_U+z^2/(2n), p̂_U^2
11 conf_int_wilson_corrected = [(b_L - √(b_L^2-a*c_L))/a, (b_U + √(b_U^2-a*c_U))/a]
12
13 p_value_wilson = 2ccdf(Normal(), abs(p̂ - p)/SE)
14 a, b, c = 1+z^2/n, p̂+z^2/(2n), p̂^2
15 conf_int_wilson = [(b - √(b^2-a*c))/a, (b + √(b^2-a*c))/a]
16
17 println()
18 @show p_value_wilson_corrected
19 @show conf_int_wilson_corrected
20 println()
21 @show p_value_wilson
22 @show conf_int_wilson
23 ;
```

```
n = 20
k = 10
p = 0.72
α = 0.05
```

```
z = quantile(Normal(), 1 - α / 2) = 1.9599639845400576
p̂ = k / n = 0.5
SE = √((p * (1 - p)) / n) = 0.10039920318408907
(p̂_L, p̂_U) = ((k - 0.5) / n, (k + 0.5) / n) = (0.475, 0.525)
```

```
p_value_wilson_corrected = 0.052107267466663096
conf_int_wilson_corrected = [0.278536702420732, 0.7214632975792679]
```

```
p_value_wilson = 0.028433528913982754
conf_int_wilson = [0.299298008198212, 0.7007019918017879]
```

4.5 Pearsonの χ^2 検定との関係

標準正規分布に従う確率変数の2乗は自由度 1 の χ^2 分布に従うことを使えば、上のP値は次のように表される:

$$pvalue_{\text{Wilson}}(k|n, p = p_0) = 1 - \text{cdf}\left(\text{Chisq}(1), \frac{(k - np_0)^2}{np_0(1 - p_0)}\right).$$

このP値を、スコア統計量(score statistics)に基くP値と呼んだり、Pearsonの χ^2 統計量

$$\frac{(k - np_0)^2}{np_0(1 - p_0)} = \frac{(k - np_0)^2}{np_0} + \frac{((n - k) - n(1 - p_0))^2}{n(1 - p_0)}$$

に基くP値と呼んだりする. 一般にPearsonの χ^2 統計量は

$$(\text{Pearsonの}\chi^2\text{統計量}) = \sum \frac{(\text{観測値} - \text{帰無仮説下の統計モデル内での期待値})^2}{\text{帰無仮説下の統計モデル内での期待値}}$$

のスタイルで定義される. 上の場合には、観測値が成功回数 k と失敗回数 $n - k$ であり、対応する帰無仮説下の統計モデル内での期待値はそれぞれ np_0 , $n(1 - p_0)$ である.

注意: Pearsonの χ^2 統計量は多項分布モデルでの検定や分割表の独立性検定でも大活躍することになる. 興味がわいた人はおまけの解説も参照せよ.

4.6 おまけ: 多項分布モデルにおけるPearsonの χ^2 検定

Pearsonの χ^2 統計量は多項分布モデルでも利用できる.

例えば、検定したい仮説「A, B, Cが生じる確率はそれぞれ $p_{A,0}, p_{B,0}, p_{C,0}$ である」($p_{A,0} + p_{B,0} + p_{C,0} = 1$)のとき、「 n 回中 A, B, C がそれぞれ a, b, c 回生じた」というデータに対して、Pearsonの χ^2 統計量を

$$(\text{Pearsonの}\chi^2\text{統計量}) = \frac{(a - np_{A,0})^2}{np_{A,0}} + \frac{(b - np_{B,0})^2}{np_{B,0}} + \frac{(c - np_{C,0})^2}{np_{C,0}}$$

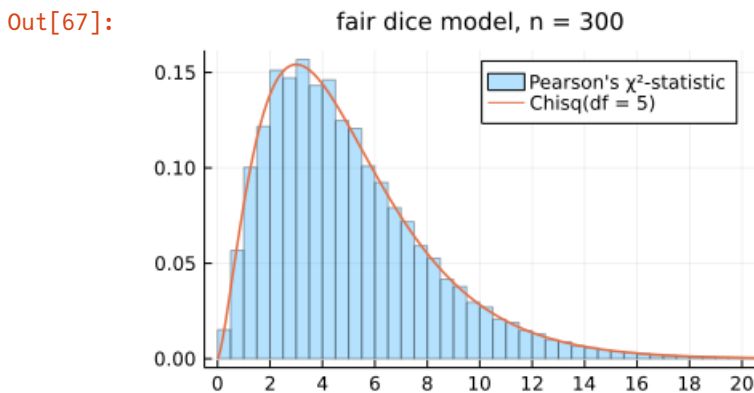
で定義できる. 二項分布の中心極限定理の拡張になっている多項分布の中心極限定理によって, このPearsonの χ^2 統計量は上の仮説下の多項分布内で自由度 2 の χ^2 分布に近似的に従うことを示せる.

このことを使えば上と同様の方法で多項分布モデルの場合にもP値を定義できる. 例えば, 出目に偏りがあることが疑われているサイコロを n 回ふったら $i = 1, 2, \dots, 6$ の目が a_i 回出たとき, 仮説「すべての目が等確率で出る」に関するPearsonの χ^2 統計量は

$$(\text{Pearsonの}\chi^2\text{統計量}) = \sum_{i=1}^6 \frac{(a_i - n/6)^2}{n/6}$$

になる. その仮説下の多項分布内でこれは, n が大きいければ, 自由度 5 の χ^2 分布に近似的に従う. このときP値は, 自由度 5 の χ^2 分布において値がPearsonの χ^2 統計量以上になる確率として定義される.

```
In [67]: 1 n, p = 300, [1/6, 1/6, 1/6, 1/6, 1/6, 1/6]
2 null = Multinomial(n, p) # fair dice model
3 df = ncategories(null) - 1 # degree of freedom
4
5 chisq_multinomial(a, p, n=sum(a)) = sum((a - n*p)^2/(n*p) for (a, p) in zip(a, p))
6 tmp = zeros(ncategories(null))
7 chisqs = [chisq_multinomial(rand!(null, tmp), p) for _ in 1:10^6]
8
9 histogram(chisqs; norm=true, alpha=0.3, bin=0:0.5:40,
10 label="Pearson's  $\chi^2$ -statistic")
11 plot!(Chisq(df); label="Chisq(df = $df)", lw=1.5)
12 title!("fair dice model, n = $n")
13 plot!(); xlim=(-0.5, 20.5), xtick=0:2:30)
```



例えば, 有意水準を $\alpha = 0.1\%$ としたとき, $n = 300$ 回中 1, 2, ..., 6 が出た回数がそれぞれ 75, 31, 43, 45, 47, 60 ならば, 仮説「すべての目が同じ確率で出る」について, Pearsonの χ^2 統計量の値は 23.38 になり, 自由度 5 の χ^2 分布でそれ以上の値になる確率として定義されたP値は 0.03% 未満になるので, 「すべての目が同じ確率で出る」という仮説は有意水準 $\alpha = 0.1\%$ で棄却されることになる.

```
In [68]: 1 a = [75, 31, 43, 45, 47, 60]
2 @show c = chisq_multinomial(a, p)
3 pvalue_multinomial = ccdf(Chisq(5), c)
4 @show pvalue_multinomial;

c = chisq_multinomial(a, p) = 23.29900332225914
pvalue_multinomial = 0.0002959051506978015
```

4.7 おまけ: 一般の χ^2 検定で使う χ^2 分布の自由度

一般の χ^2 検定で使用する χ^2 分布の自由度 df (degree of freedom)はWilks' theorem (https://en.wikipedia.org/wiki/Wilks%27_theorem)より次の公式で決定される:

$$\begin{aligned} \text{df} &= (\text{統計モデルの独立なパラメータの個数}) - (\text{帰無仮説下の統計モデルの独立なパラメータ個数}) \\ &= (\text{帰無仮説によって減るモデルの独立なパラメータ個数}). \end{aligned}$$

例えば, 上のサイコロの例では, 六項分布モデルのパラメータ数は $6 - 1 = 5$ 個である. なぜならば $i = 1, 2, \dots, 6$ の各々の目が出る確率を p_i と書くとき, それらには $p_1 + p_2 + \dots + p_6 = 1$ という条件が課されるので, 独立なパラメータの個数は 5 個になる. そして, 帰無仮説「すべての目が等確率で出る」を課するとすべての p_i の値が $1/6$ に固定されて, 独立に動けるパラメータの個数は 0 になる. 帰無仮説によって独立なパラメータの個数が 5 個減ったので, χ^2 検定で使う χ^2 分布の自由度は 5 になる.

例えば, 帰無仮説が

$$p_1 + p_6 = 1/3, \quad p_2 + p_5 = 1/3, \quad p_3 + p_4 = 1/3$$

と書ける. これら3本のうち, 条件 $p_1 + p_2 + \dots + p_6 = 1$ の下で独立な方程式は2本なので, この仮説を課すことによって独立なパラメータの個数は 2 個減る. ゆえに, この場合には, χ^2 検定で使用する χ^2 分布の自由度は 2 になる. ただし, この場合には

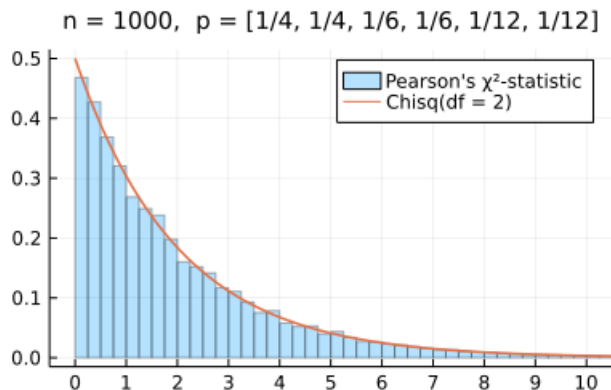
$$(\text{Pearsonの}\chi^2\text{統計量}) = \sum \frac{(\text{観測値} - \text{帰無仮説下の統計モデル内での期待値})^2}{\text{帰無仮説下の統計モデル内での期待値}}$$

における「帰無仮説下の統計モデル内での期待値」は統計モデルのパラメータを帰無仮説の制限の下での最尤法で決定することによって求めることになる.

In [69]:

```
1 n, p = 1000, [1/4, 1/4, 1/6, 1/6, 1/12, 1/12]
2 null = Multinomial(n, p) # P(odd) = P(even)
3 df = 2 # degree of freedom
4
5 function chisq_multinomial_example2(a)
6     n = sum(a)
7     r = length(a)
8     @assert r == 6
9     mle(i) = if i == 1 || i == 6
10         a[i]/(3(a[1] + a[6]))
11     elseif i == 2 || i == 5
12         a[i]/(3(a[2] + a[5]))
13     else
14         a[i]/(3(a[3] + a[4]))
15     end
16     sum((a[i] - n*mle(i))^2/(n*mle(i)) for i in 1:r)
17 end
18 tmp = zeros(ncategories(null))
19 chisqs = [chisq_multinomial_example2(rand!(null, tmp)) for _ in 1:10^6]
20
21 histogram(chisqs; norm=true, alpha=0.3, bin=0:0.25:20,
22     label="Pearson's  $\chi^2$ -statistic")
23 plot!(Chisq(df); label="Chisq(df = $df)", lw=1.5)
24 title!("n = $n, p = [1/4, 1/4, 1/6, 1/6, 1/12, 1/12]")
25 plot!(); xlim=(-0.5, 10.5), xtick=0:20)
```

Out[69]:



例えば, 有意水準を $\alpha = 0.1\%$ としたとき, $n = 1000$ 回中 1, 2, ..., 6 が出た回数がそれぞれ 235, 195, 175, 175, 195, 225 ならば, 仮説

$$p_1 + p_6 = 1/3, \quad p_2 + p_5 = 1/3, \quad p_3 + p_4 = 1/3$$

について, Pearsonの χ^2 統計量の値は 15.5 になり, 自由度 2 の χ^2 分布でそれ以上の値になる確率として定義されたP値は 0.05% 未満になるので, 上の仮説は有意水準 $\alpha = 0.1\%$ で棄却されることになる.

In [70]:

```
1 a = [235, 195, 175, 175, 195, 225]
2 @show c = chisq_multinomial_example2(a)
3 pvalue_multinomial_example2 = ccdf(Chisq(2), c)
4 @show pvalue_multinomial_example2;

c = chisq_multinomial_example2(a) = 15.5
pvalue_multinomial_example2 = 0.00043074254057568753
```



```
In [71]: 1 [1/4, 1/4, 1/6, 1/6, 1/12, 1/12] ► sum
```

```
Out[71]: 1.0
```

5 Wald

5.1 Waldの信頼区間を与えるP値関数

\hat{p}, \widehat{SE} を以下のように定義する:

$$\hat{p} = \frac{k}{n}, \quad \widehat{SE} = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}.$$

データ「 n 回中 k 回成功」について、「データの数値以上に極端な」の意味を「 k 以上または k 以下の」とした場合. ただし, 二項分布の中心極限定理(正規分布近似)と分散を推定値に置き換える粗い近似を使う:

$$pvalue_{\text{Wald}}(k|n, p = p_0) = 2 \left(1 - \text{cdf} \left(\text{Normal}(0, 1), \frac{|k - np_0|}{\sqrt{n\hat{p}(1-\hat{p})}} \right) \right).$$

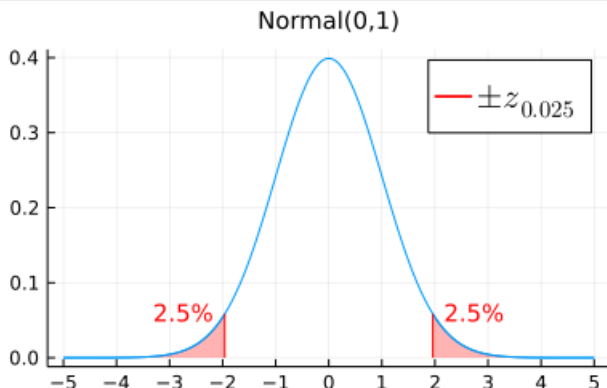
$pvalue_{\text{Wilson}}(k|n, p = p_0)$ との違いは右辺の cdf の引数の分母の p_0 がその推定値 \hat{p} に置き換えられていることである. これは粗い近似になっているが, その分だけ信頼区間の計算を易しくしてくれる.

5.2 Waldの信頼区間

$z_{\alpha/2} = z_{1-\alpha/2} = \text{quantile}(\text{Normal}(0, 1), 1 - \alpha/2)$ とおく.

```
In [72]: 1 var"z_{0.025} の定義"
```

```
Out[72]:
```



この場合の信頼区間 $\{p \in [0, 1] \mid pvalue_{\text{Wald}}(k|n, p) \geq \alpha\}$ は次のように表される:

$$\text{confint}_{\text{Wald}}(k|n, p = p_0) := \left[\hat{p} - z_{\alpha/2} \widehat{SE}, \hat{p} + z_{\alpha/2} \widehat{SE} \right].$$

これを **Waldの信頼区間** と呼ぶ.

前節の公式と比較すれば, n を大きくすると, これはWilsonの信頼区間をよく近似するようになることがわかる.

Waldの信頼区間の導出については, [「例: ベータ関数と二項分布の関係とその応用」のノート](#)

([https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/07-](https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/07-1%20Relationship%20between%20beta%20and%20binomial%20distributions.ipynb)

[1%20Relationship%20between%20beta%20and%20binomial%20distributions.ipynb](https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/07-1%20Relationship%20between%20beta%20and%20binomial%20distributions.ipynb))の「Waldの信頼区間の基礎」の節も参照せよ.

5.3 Waldの信頼区間の図による把握

以下の図をよく見れば, $\alpha = 5\%$ の場合の $p_L = p_{-L}$ と $p_U = p_{+U}$ を直観的に把握できるだろう.

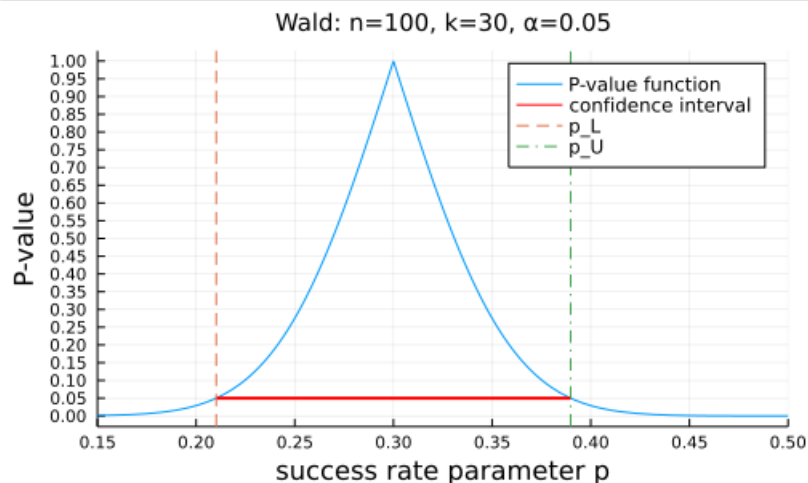
まず, データの数値が与えられたとき, モデルのパラメータ値をP値に対応させる **P値関数** (P-value function)の立場では, **信頼区間** (confidence interval)の解釈は易しい.

信頼区間はP値関数が有意水準の値を取る2つのパラメータに挟まれた区間になる.

しかし、P値函数全体のグラフを描けば、信頼区間を計算しなくても、データとの整合性が低過ぎないパラメータ値の範囲は明らかだろう。

```
In [73]: 1 plot_binom_pvalue_and_confint(  
2     pvalue_wald, confint_wald, 100, 30;  
3     name = "Wald: ", xlim=(0.15, 0.5), xtick=0:0.05:1)
```

Out[73]:

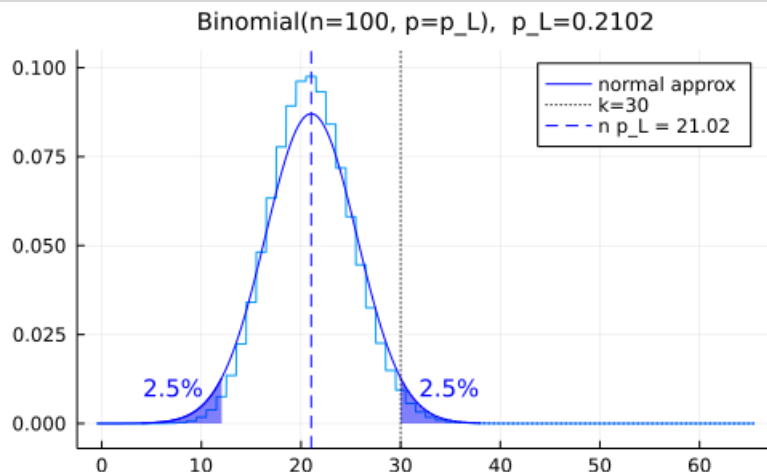


$1 - \alpha = 95\%$ 信頼区間の下限を求めるためには、二項分布モデルの成功確率パラメータ p を小さくして行って、その二項分布を近似する正規分布内でデータの数値 k 以上の値が生成される確率とその反対側の確率の和 (= Waldの信頼区間を与えるP値)がちょうど $\alpha = 5\%$ になるような p を見つければよい。そのような p を $p_L = p_{\text{L}}$ とする。

ただし、Waldの信頼区間の場合にはWilsonの信頼区間の場合と違って、二項分布を正規分布で近似するときには、正規分布の分散をデータの数値から推定したものに固定する。そのことが原因で二項分布の正規分布近似の誤差は大きくなる。

```
In [74]: 1 var"Wald: 下側では p = p_L がぎりぎり"
```

Out[74]:

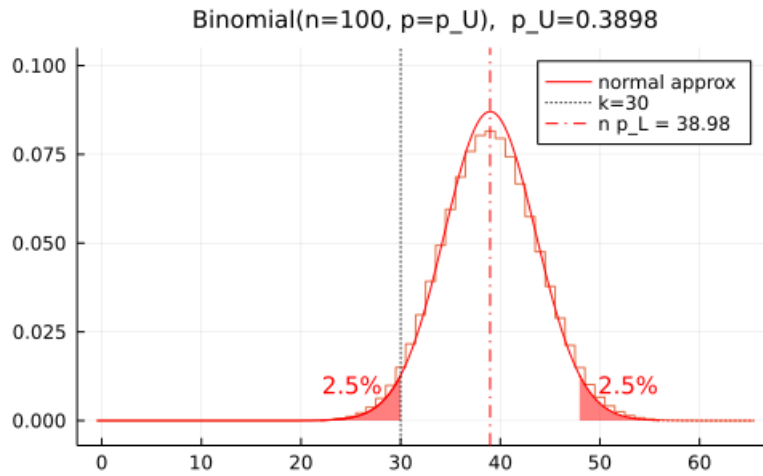


$1 - \alpha = 95\%$ 信頼区間の上限を求めるためには、二項分布モデルの成功確率パラメータ p を大きくして行って、その二項分布を近似する正規分布内でデータの数値 k 以下の値が生成される確率とその反対側の確率の和 (= Waldの信頼区間を与えるP値)がちょうど $\alpha = 5\%$ になるような p を見つければよい。そのような p を $p_U = p_{\text{U}}$ とする。

ただし、上でも説明したように、Waldの信頼区間の場合にはWilsonの信頼区間の場合と違って、二項分布を正規分布で近似するときには、正規分布の分散をデータの数値から推定したものに固定する。そのことが原因で二項分布の正規分布近似の誤差は大きくなる。

In [75]: 1 var"Wald: 上側では $p = p_U$ がぎりぎり"

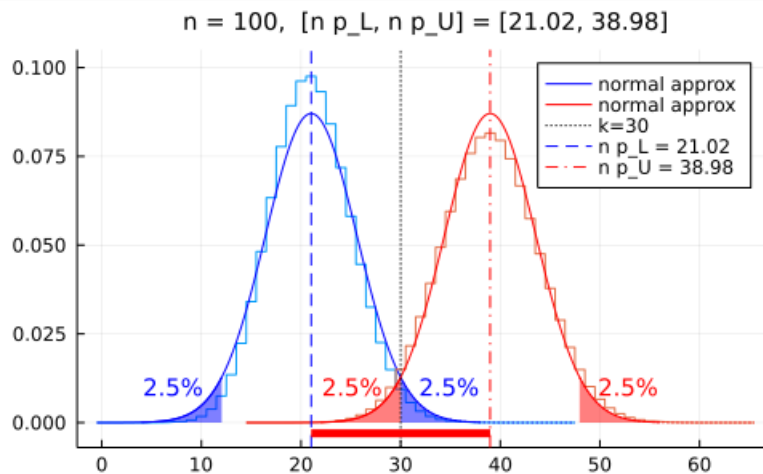
Out[75]:



以上をまとめると以下のような図を描ける.

In [76]: 1 var"Wald: $n \times 95\%$ 信頼区間"

Out[76]:



この場合のWaldの信頼区間は $[0.2102, 0.3998]$ である.

これは、同じ場合のClopper-Pearsonの信頼区間の $[0.2124, 0.3998]$ とSterneの信頼区間は $[0.2142, 0.3996]$ とWilsonの信頼区間は $[0.2189, 0.3958]$ のどれよりも少し広がっているが、ほぼ同じであると言ってよいだろう.

ただし、Waldの信頼区間は \hat{p} または $1 - \hat{p}$ が小さい場合には誤差が大きくなる場合があるので注意が必要である.

In []: 1