

まとめ

- 黒木玄
- 2022-07-20～2022-07-23

このノートではJulia言語 (<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。膨大な時間を取り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 二項分布モデルでのClopper-Pearsonの信頼区間](#)
 - [1.1 Clopper-Pearsonの信頼区間を例に信頼区間の解釈の仕方について説明](#)
 - [1.2 二項分布モデルのClopper-Pearsonの信頼区間の効率的な計算の仕方](#)
 - [1.3 二項分布モデルのBayes統計](#)
 - [1.4 P値とBayes統計の関係](#)
 - ▼ [1.5 復習: 二項分布とベータ分布の関係の証明](#)
 - [1.5.1 両辺を \$p\$ で微分して確認する方法](#)
 - [1.5.2 ベータ分布の一様分布の順序統計量の分布として解釈を使う方法](#)
- ▼ [2 P値函数について](#)
 - [2.1 P値はモデルのパラメータ値とデータの数値の整合性の指標の1つ](#)
 - [2.2 P値は信頼区間を考えたいすべてのパラメータ値について定義されている](#)
 - [2.3 分割表の場合のP値函数](#)
 - [2.4 P値函数から信頼区間が定義される](#)
 - [2.5 2×2の分割表のP値と信頼区間とP値函数の例](#)
 - [2.6 2×2の分割表での検定ではどれを使うべきか](#)
 - [2.7 P値函数と最尤法の関係](#)
 - [2.8 以上のまとめの図](#)
 - [2.9 おまけ: Bayes統計の方法を使った場合](#)
- ▼ [3 Welchのt検定について](#)
 - [3.1 Welchのt検定のP値と信頼区間の定義](#)
 - ▼ [3.2 Welchのt検定のP値や信頼区間の計算例](#)
 - [3.2.1 WolframAlphaによるWelchのt検定のP値と信頼区間の計算の必修問題の解答例](#)
 - [3.2.2 Julia言語によるWelchのt検定のP値と信頼区間の計算の必修問題の解答例](#)
- ▼ [4 数学的な補足: 大数の法則と中心極限定理について](#)
 - [4.1 二項分布の大数の法則](#)
 - [4.2 二項分布の中心極限定理](#)
 - ▼ [4.3 他の分布の場合](#)
 - [4.3.1 例: Poisson分布](#)
 - [4.3.2 例: Gamma分布](#)
 - [4.4 標本分布の場合](#)
 - [4.5 再掲: 大数の法則と中心極限定理のイメージ](#)
 - [4.6 統計学の基礎になる確率論の三種の神器](#)
 - [4.7 Kullback-Leibler情報量に関するSanovの定理の数値例](#)

In [1]:

```

1 ENV["LINES"], ENV["COLUMNS"] = 100, 100
2 using Base.Threads
3 using BenchmarkTools
4 using DataFrames
5 using Distributions
6 using LinearAlgebra
7 using Memoization
8 using Optim
9 using Printf
10 using QuadGK
11 using RCall
12 @import stats as R
13 using Random
14 Random.seed!(4649373)
15 using Roots
16 using SpecialFunctions
17 using StaticArrays
18 using StatsBase
19 using StatsFuns
20 using StatsPlots
21 default(fmt = :png, size = (400, 250),
22         titlefontsize = 10, guidefontsize=9, plot_titlefontsize = 12)
23 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 " *
6                           sympy.latex(x, mode="plain", fold_short_frac=false)))
7 end
8 @eval SymPy function Base.show(io::IO, ::MIME"text/latex", x::AbstractArray{Sym})
9     function toeqnarray(x::Vector{Sym})
10        a = join(["\"\\displaystyle " *
11                   sympy.latex(x[i]) for i in 1:length(x)], "\\\\\\")
12        """\"\\left[ \"\\begin{array}{r} a \"\\end{array} \"\\right]"""
13    end
14    function toeqnarray(x::AbstractArray{Sym,2})
15        sz = size(x)
16        a = join([join("\\"displaystyle " .* map(sympy.latex, x[i,:]), "&")
17                  for i in 1:sz[1]], "\\\\\\")
18        "\"\\left[ \"\\begin{array}{\" * repeat("r",sz[2]) * "}" * a * "\"\\end{array} \"\\right]"
19    end
20    print(io, as_markdown(toeqnarray(x)))
21 end

```

In [3]:

```

1 safemul(x, y) = x == 0 ? x : isinf(x) ? typeof(x)(Inf) : x*y
2 safediv(x, y) = x == 0 ? x : isinf(y) ? zero(y) : 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}) =
17     standardized_moment(dist, 3)
18 mykurtosis(dist::MixtureModel{Univariate, Continuous}) =
19     standardized_moment(dist, 4) - 3

```

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

In [4]:

```

1 function logtick(; xlim=(0.03, 30))
2     xmin, xmax = xlim
3     a = floor(Int, log10(xmin))
4     b = ceil(Int, log10(xmax))
5     nums = [1, 2, 3, 4, 5, 6, 7, 8, 9]
6     mask = Bool[1, 1, 0, 0, 1, 0, 0, 0, 0]
7
8     logtick = foldl(vcat, ([10.0^k*x for x in nums if xmin ≤ 10.0^k*x ≤ xmax] for k in a:b)
9     logticklabel_a = foldl(vcat,
10         ([mask[i] ? string(round(10.0^k*x; digits=-k)) : ""
11             for (i, x) in enumerate(nums) if xmin ≤ 10.0^k*x ≤ xmax]
12             for k in a:-1))
13     logticklabel_b = foldl(vcat,
14         ([mask[i] ? string(10^k*x) : ""
15             for (i, x) in enumerate(nums) if xmin ≤ 10.0^k*x ≤ xmax]
16             for k in 0:b))
17     logticklabel = vcat(logticklabel_a, logticklabel_b)
18     (logtick, logticklabel)
19 end
20
21 # logtick()

```

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

1 二項分布モデルでのClopper-Pearsonの信頼区間

1.1 Clopper-Pearsonの信頼区間を例に信頼区間の解釈の仕方について説明

「 n 回中 k 回当たりが出た」 「 n 人中 k 人が重症化」 「 n 人中 k 人が商品を購入」などの型のデータを扱うための統計モデルとして、成功確率パラメータ p を持つ二項分布

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

を使う状況を考える。このとき、仮説 $p = p_0$ のP値(データの数値とモデル+パラメータ値の整合性の指標の1つ)を次のように定義できるのであった:

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

これをClopper-PearsonのP値と呼ぶことにしていた。このP値は以下の3つのうちの最小値として定義されている:

- 1
- 仮説 $p = p_0$ の下での二項分布で n 回中の当たりの回数が k 回以下の確率の2倍
- 仮説 $p = p_0$ の下での二項分布で n 回中の当たりの回数が k 回以上の確率の2倍

本当はモデル内で仮想的に生成されたデータの値 i (n 回中の当たりの回数)が期待値 np_0 からデータの数値 k 以上離れる確率をP値としたいのだが(P値の定義は仮説下のモデル内でデータの数値以上に極端な値が生成される確率またはその近似値であった), 反対側の確率を自然に決めることができないので, 2倍して反対側の確率も足し上げたことにしている。(これは1つの処方箋に過ぎず, 別の方法もあるのであった。)

これに対応する信頼度 $1 - \alpha$ の信頼区間の定義は次の通り:

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

仮説 $p = p_0$ のP値 $\text{pvalue}_{\text{CP}}(k|n, p = p_0)$ はデータの数値「 n 回中 k 回」とモデルのパラメータに関する仮説 $p = p_0$ の整合性の指標として使われるのであった。

信頼区間を定義している条件 $\text{pvalue}_{\text{CP}}(k|n, p = p_0) \geq \alpha$ はその整合性の指標の値がある閾値 α (この閾値は有意水準と呼ばれる)以上になっているという条件になっている。

通常, 有意水準 α は目的に合わせて小さな値を採用し, 検定の手続きではP値が α 未満になるパラメータ値 $p = p_0$ を棄却する。

信頼度 $1 - \alpha$ 信頼区間は有意水準 α で棄却されないパラメータ値 $p = p_0$ 全体の集合になっている。

「棄却されないこと」は「判断を保留すること」を意味する。

信頼区間に含まれるパラメータ値 $p = p_0$ については判断を保留することになるが, もしも信頼区間が十分に狭くなってしまえば, その外側にある大部分のパラメータ値は検定の手続きによって棄却されており, 可能性として考慮すればよいのは, 十分に狭くなっている信頼区間に含まれるパラメータ値だけになっているので, 科学的に十分に意味のある結論を出すことができる

場合が出来ます。

ただし、信頼区間を使って科学的に信頼できる結論を出すためには、データの取得法が信頼できるか、採用した統計モデルは妥当であるか、その他見落としている重要な問題はないか、などの多くの問題をクリアする必要がある。

機械的に信頼区間を計算して科学的な御黒付きが得られたような態度を取るアトは誤りなので注意して下さい。

1.2 二項分布モデルのClopper-Pearsonの信頼区間の効率的な計算の仕方

Clopper-Pearsonの信頼区間は定義をうまく変形すると以下のように書き直される：

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

ここで、 p_L, p_U は次の条件で特徴付けられる値である：

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

すなわち、小さい方の p_L は $p = p_L$ の二項分布内で n 回中の当たりの回数が k 以上になる確率が $\alpha/2$ になるという条件で特徴付けられ、大きい方の p_U は $p = p_U$ の二項分布内で n 回中当たりの回数が k 回以下になる確率が $\alpha/2$ になるという条件で特徴付けられる。

問題はこの p_L, p_U をどのように計算するかである。

これには二項分布とベータ分布の間の関係を使う驚くべき方法がある。

二項分布の累積分布函数(cumulative distribution function, cdf)の定義は

$$\text{cdf}(\text{Binomial}(n, p), k) = \sum_{i=0}^k \binom{n}{i} p^i (1-p)^{n-i}$$

である。場合によっては万単位の個数の値を足し上げる計算になる。

しかし、そういう和を取る計算をコンピュータで素朴に行うのはコンピュータ資源の無駄遣いになってしまふ。

なぜならば、すでに説明したように非常にうまい方法があるからである。

二項分布とベータ分布の関係：

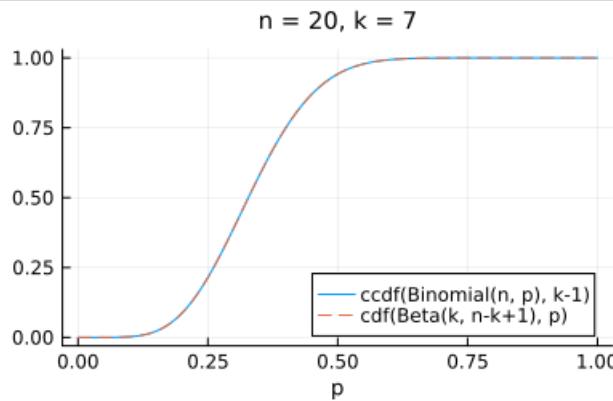
$$1 - \text{cdf}(\text{Binomial}(n, p), k-1) = \text{ccdf}(\text{Binomial}(n, p), k-1),$$
$$\text{ccdf}(\text{Binomial}(n, p), k) = 1 - \text{cdf}(\text{Beta}(k+1, n-k), p).$$

この前者と後者の公式は同値な公式になっており、片方からもう一方が導かれる。

以下でこれらの前者を $n = 20, k = 7$ の場合に数値的に確認してみよう。

```
In [5]: 1 n, k = 20, 7
2 plot(p → ccdf(Binomial(n, p), k-1), 0, 1; label="ccdf(Binomial(n, p), k-1)")
3 plot!(p → cdf(Beta(k, n-k+1), p); label="cdf(Beta(k, n-k+1), p)", ls=:dash)
4 title!("n = $n, k = $k", xguide="p", legend=:bottomright)
```

Out[5]:



2つのグラフがぴったり重なり合っている。(ccdf は $1 - \text{cdf}$ の意味である。)

ベータ分布の累積分布函数 $\text{cdf}(\text{Beta}(k+1, n-k), p)$ は正則化された不完全ベータ函数という名前がついている基本特殊函数になっており、その p に関する逆函数(分位点函数, quantile function)もコンピュータの基本特殊函数ライブラリの中で効率的に実装されている。だから、

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

を満たす p_L は、ベータ分布の分位点函数(quantile function)を用いて、

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

を使えばコンピュータで効率的に計算できる。同様に

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

を満たす p_U は

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

を使えばコンピュータで効率的に計算できる。

まとめ: 二項分布モデルでのClopper-Pearsonの信頼区間 $[p_L, p_U]$ は

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

と計算できる。

二項分布とベータ分布の関係が、コンピュータ上に実装された基本特殊函数ライブラリを通して、二項分布モデルの場合の信頼区間の効率的な計算に役に立っている！

- 異なる確率分布に間の関係に関する数学
- 不完全ベータ函数などの基本特殊函数に関する数学
- それをコンピュータで実装するプログラミングの技術

などの多くの技術を組み合わせることによって、Clopper-Pearsonの信頼区間のシンプルで効率的な計算法が実現されている！

このようなことから、**このClopper-Pearsonの信頼区間についてこのように理解することは、高等教育を受けた人達にとっての重要な教養になり得る** と思われる。

例えは、 $n = 20, k = 7, \alpha = 0.05$ の場合の信頼度 $1 - \alpha$ のClopper-Pearsonの信頼区間は次のように計算される。

```
In [6]: 1 n, k, α = 20, 7, 0.05
2 [quantile(Beta(k, n-k+1), α/2), quantile(Beta(k+1, n-k), 1-α/2)] ▷ println
```

```
[0.1539092047845412, 0.5921885345328282]
```

R言語を使って計算した結果も同じになっている：

```
In [7]: 1 rcopy(R"""binom.test(7, 20, p=0.05)""")[:conf_int] ▷ println
```

```
[0.15390920478454115, 0.5921885345328282]
```

WolframAlphaでも以下のようにすれば容易に計算できる。

$p_L \rightarrow \text{quantile}(\text{BetaDistribution}(7, 20-7+1), 0.025) \rightarrow [\text{実行}(\text{https://www.wolframalpha.com/input?i=quantile%28BetaDistribution%287%2C+20-7%2B1%29%2C+0.025%29})] \rightarrow 0.153909$

$p_U \rightarrow \text{quantile}(\text{BetaDistribution}(7+1, 20-7), 0.975) \rightarrow [\text{実行}(\text{https://www.wolframalpha.com/input?i=quantile%28BetaDistribution%287%2B1%2C+20-7%29%2C+0.975%29})] \rightarrow 0.592189$

1.3 二項分布モデルのBayes統計

二項分布とベータ分布の関係は通常の検定とBayes統計の関係を理解するために役に立つ。

二項分布のベイズ統計では先に用意した二項分布の確率質量函数 $P(k|n, p)$ だけではなく、パラメータ p に関する確率密度函数 $\varphi(p)$ を任意に与え(目的に合わせて適切に事前分布を決めることがあるが、理論的には任意でよい)、 (k, p) に関する同時分布 (joint distribution)

$$P(k, p|n) = P(k|n, p)\varphi(p), \quad \sum_{k=0}^n \int_0^1 P(k, p|n) dp = 1$$

を考える。そして、「 n 回中 k 回」の形のデータの数値が与えられたとき、このモデルの同時確率分布を、データと同じ数値が生成されたという条件で制限して得られるパラメータ p に関する条件付き確率分布 $\varphi(p|n, k)$ を考える：

$$\varphi(p|n, k) = \frac{P(k|n, p)\varphi(p)}{\int_0^1 P(k|n, p)\varphi(p) dp}.$$

このとき、 $\varphi(p)$ を**事前分布** (prior)と呼び、 $\varphi(p|n, k)$ を**事後分布** (posterior)と呼ぶ。

条件付き確率分布については

- ・「[条件付き確率分布, 尤度, 推定, 記述統計](#)」のノート

(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/06%20Conditional%20distribution%2C%20likelihood%2C.ipynb>)

に解説がある。数学的には

- ・確率 = 全体に対する部分の割合
- ・条件付き確率 = 部分に対する部分の割合

のように考えると直観的に理解し易いと思う。上の事後分布の公式を **ベイズの定理** と呼ぶこともあるが、ベイズの定理を知らない場合、条件付き確率分布について理解していかなければ困らない。

事後分布 $\varphi(p|n, k)$ の $p = p_0$ での値 $\varphi(p = p_0|n, k)$ はデータの数値 $p = p_0$ とデータの数値 n, k の統計モデル $P(k, p|n) = P(k|n, p)\varphi(p)$ の下での整合性の指標として利用できる。(P値とはまた別のモデルとデータの整合性の指標が得られたことになる。)

さて、例によって、問題はどのように事後分布と呼ばれる条件付き確率分布を計算するかである。

事後分布の公式を見れば分母が積分

$$Z(k|n) := \int_0^1 P(k|n, p)\varphi(p) dp$$

になっていて、その積分の部分の計算が大変そうなことがわかる。(事後分布の分母を **周辺尤度** (marginal likelihood)と呼ぶことがある。他にも様々な呼び方があるが、呼び方自体は重要ではない。)

二項分布の場合には、事前分布としてベータ分布を採用すると、事後分布もベータ分布になり、事後分布の計算が著しく単純化される。この事実をベータ分布は二項分布の **共役事前分布** (conjugate prior)であるという。

以下では $\varphi(p)$ が Beta(a, b) ($a, b > 0$) の確率密度函数であると仮定する:

$$\varphi(p) = \frac{p^{a-1}(1-p)^{b-1}}{B(a, b)} \quad (0 < p < 1).$$

このとき、事後分布の分母は次のように計算される:

$$\begin{aligned} Z(k|n) &= \int_0^1 \binom{n}{k} p^k (1-p)^{n-k} \frac{p^{a-1}(1-p)^{b-1}}{B(a, b)} dp \\ &= \binom{n}{k} \frac{1}{B(a, b)} \int_0^1 p^{a+k-1} (1-p)^{b+n-k} dp \\ &= \binom{n}{k} \frac{B(a+k, b+n-k)}{B(a, b)}. \end{aligned}$$

ゆえに、事後分布は次のようになる:

$$\begin{aligned} \varphi(p|n, k) &= \frac{P(k|n, p)\varphi(p)}{Z(k|n)} \\ &= \binom{n}{k} \frac{1}{B(a, b)} p^{a+k-1} (1-p)^{b+n-k} \times \left(\binom{n}{k} \frac{B(a+k, b+n-k)}{B(a, b)} \right)^{-1} \\ &= \frac{p^{a+k-1} (1-p)^{b+n-k}}{B(a+k, b+n-k)}. \end{aligned}$$

以上によって、事前分布が共役事前分布 Beta(a, b) のとき、データ「 n 回中 k 回」から得られる事後分布は Beta($a + k, b + n - k$) になることがわかった。

$a, b > 0$ と仮定していたが、 $a = 0$ または $b = 0$ の場合にも、 $a + k$ と $b + n - k$ が共に正になるならば、事後分布は Beta($a + k, b + n - k$) はうまく定義されている。このような場合のうまく定義されていない事前分布は **improper事前分布** (improper prior)と呼ばれている。

この節の事柄については

- ・「[例：ベータ函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>)

1.4 P値とBayes統計の関係

二項分布とベータ分布の累積分布函数の間には以下の関係があるのであった:

$$1 - \text{cdf}(\text{Binomial}(n, p_0), k - 1) = \text{cdf}(\text{Beta}(k, n - k + 1), p_0),$$

$$\text{cdf}(\text{Binomial}(n, p_0), k) = 1 - \text{cdf}(\text{Beta}(k + 1, n - k), p_0).$$

一般に分布 D について累積分布函数 $\text{cdf}(D, x)$ は分布 D に従う確率変数 X が x 以下になる確率を意味することに注意せよ.

ゆえに上の公式の前者の両辺は以下のような解釈を持つ:

- $1 - \text{cdf}(\text{Binomial}(n, p_0), k - 1)$ (分布 $\text{Binomial}(n, p_0)$ で k 以上になる確率) は仮説 $p \leq p_0$ に関する片側検定のP値である.
- $\text{cdf}(\text{Beta}(k, n - k + 1), p_0)$ はimproper事前分布 $\text{Beta}(0, 1)$ の下での事後分布 $\text{Beta}(k, n - k + 1)$ において仮説 $p \leq p_0$ が成立する確率である.

このように、仮説 $p \leq p_0$ の片側検定のP値はBayes統計での事後分布において仮説 $p \leq p_0$ が成立する確率に一致する. 同様に,

- $\text{cdf}(\text{Binomial}(n, p_0), k)$ (分布 $\text{Binomial}(n, p_0)$ で k 以下になる確率) は仮説 $p \geq p_0$ に関する片側検定のP値である.
- $1 - \text{cdf}(\text{Beta}(k + 1, n - k), p_0)$ はimproper事前分布 $\text{Beta}(1, 0)$ の下での事後分布 $\text{Beta}(k + 1, n - k + 1)$ において仮説 $p \geq p_0$ が成立する確率である.

ゆえに、仮説 $p \geq p_0$ の片側検定のP値はBayes統計での事後分布において仮説 $p \geq p_0$ が成立する確率に一致する.

このように、数学的にはP値とBayes統計のあいだにexactな関係が付けられる場合もある.

P値を使う統計学とBayes統計は決して水と油ではない.

以上によって、片側検定のP値については、Bayes統計でのexactな解釈が得られることになる.

しかし、通常使われるP値は両側検定のP値(Clopper-Pearson型のP値では片側検定のP値の2倍を両側検定のP値として採用する)とBayes統計の関係はどうなっているのだろうか?

Clopper-Pearson型のP値のBayes統計での類似物を構成するには、上の片側検定の場合に関する結果を利用すればよい.

そのときに問題になることは、片側検定の向きを変えると、使用する(improper)事前分布が $\text{Beta}(0, 1)$ と $\text{Beta}(1, 0)$ のあいだで変化してしまうことである.

そこで、両側検定のP値のBayes統計によるexactな解釈を作ることはあきらめて、近似的な関係を得ることを目標にしよう.

そのためには事前分布として $\text{Beta}(0, 1)$ と $\text{Beta}(1, 0)$ の中間に位置する $\text{Beta}(1/2, 1/2)$ を採用すると良さそうであると予想され、実際にそうであることが計算によって確認可能である.

$\text{Beta}(1/2, 1/2)$ は二項分布の **Jeffreys事前分布** と呼ばれる.

他の選択肢として、事前分布として平坦事前分布 $\text{Beta}(1, 1)$ を採用することも考えられる。($\text{Beta}(1, 1)$ の確率密度函数は $\varphi(p) = 1$ ($0 < p < 1$) となるので、**平坦事前分布** または**一様事前分布** と呼んだりする。)

事前分布として平坦事前分布を採用すると、両側検定のP値との対応において、Jeffreys事前分布を採用した場合よりも誤差が増えるが、 k と $n - k$ が大きければ、 $\text{Beta}(k + 1/2, n - k + 1/2)$ と $\text{Beta}(k + 1, n - k + 1)$ の違いは小さくなるので、実践的な応用の場面では問題でなくなる。

以下では、事前分布が $\text{Beta}(a, a)$ である場合を考える。

このとき、事後分布は $\text{Beta}(k + a, n - k + a)$ になる。

この事後分布における仮説 $p = p_0$ の両側検定に関するClopper-Pearson型のP値函数の類似物は次のように定義される:

$$\text{pvalue}_{\text{Bayes}}(k|n, p = p_0) = \min \left(\begin{array}{c} 1 \\ 2 \text{cdf}(\text{Beta}(k + a, n - k + a), p_0) \\ 2(1 - \text{cdf}(\text{Beta}(k + a, n - k + a), p_0)) \end{array} \right).$$

これは、二項分布に関するClopper-Pearson型のP値をそのままベータ分布に一般化しただけの定義になっている。

さらに、通常のP値として、Clopper-Pearson型のP値だけではなく、正規分布近似(中心極限定理)を使ったWilson型のP値函数も考えることにしよう:

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

このとき、次が成立している。

結論: k と $n - k$ が大きければ、以上の3つのP値 $\text{pvalue}_{\text{CP}}(k|n, p = p_0)$, $\text{pvalue}_{\text{Bayes}}(k|n, p = p_0)$, $\text{pvalue}_{\text{Wilson}}(k|n, p = p_0)$ は近似的に一致する。

要するに、両側検定の通常使われるP値についても、Bayes統計での近似的な解釈が存在する。

```

1 # 上に書いてある定義の通りのP値の実装
2
3 function pvalue_cp(k, n, p)
4     bin = Binomial(n, p)
5     min(1, 2cdf(bin, k), 2ccdf(bin, k-1))
6 end
7
8 function pvalue_wilson(k, n, p)
9     z = (k - n*p)/sqrt(n*p*(1-p))
10    2ccdf(Normal(0, 1), abs(z))
11 end
12
13 function pvalue_bayes(k, n, p; a=1/2)
14     beta = Beta(k+a, n-k+a)
15     min(1, 2cdf(beta, p), 2ccdf(beta, p))
16 end
17
18 # 後で使うための別のP値の定義
19
20 function pvalue_sterne(k, n, p)
21     bin = Binomial(n, p)
22     # 次はnaiveな定義で計算効率は非常に悪い
23     sum(pdf(bin, i) for i in support(bin) if pdf(bin, i) ≤ pdf(bin, k))
24 end
25
26 function pvalue_wald(k, n, p)
27     phat = k/n
28     z = (k - n*p)/sqrt(n*phat*(1-phat))
29     2ccdf(Normal(0, 1), abs(z))
30 end

```

Out[8]: pvalue_wald (generic function with 1 method)

In [9]:

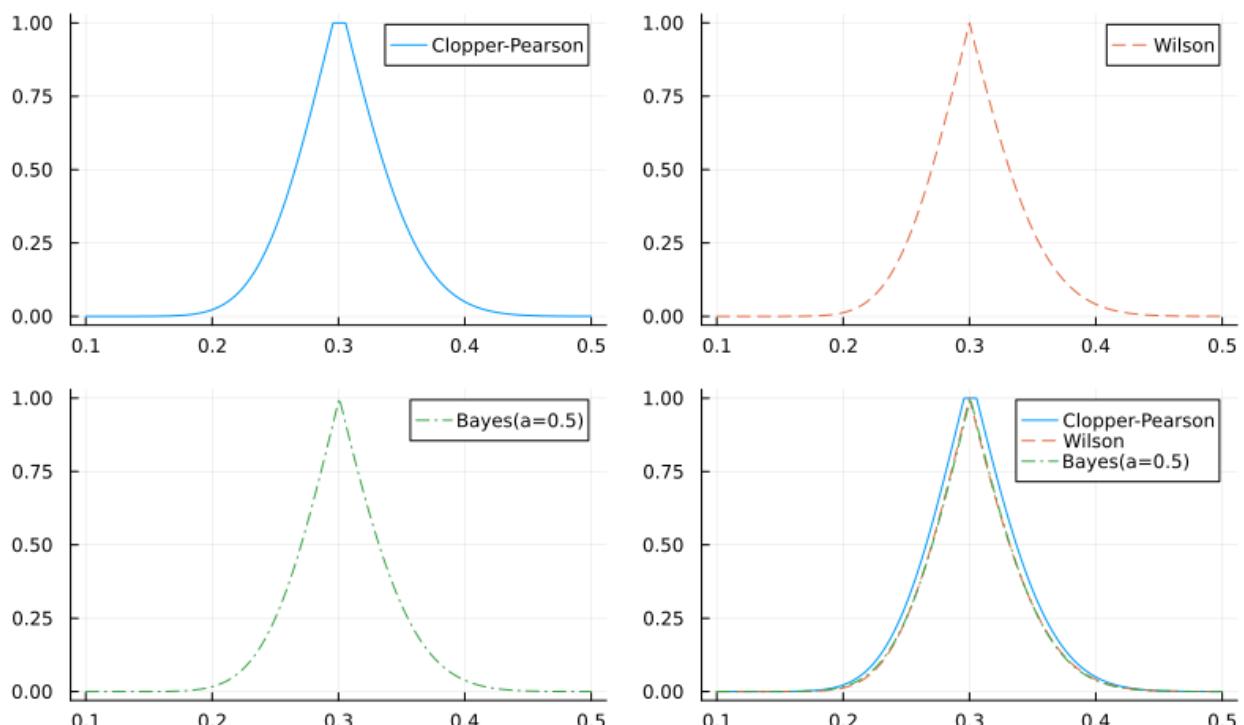
```

1 n, k, a = 100, 30, 1/2
2 xlim = (0.1, 0.5)
3 P1 = plot(p → pvalue_cp(k, n, p), xlim..., label="Clopper-Pearson")
4 P2 = plot(p → pvalue_wilson(k, n, p), xlim..., label="Wilson", c=2, ls=:dash)
5 P3 = plot(p → pvalue_bayes(k, n, p; a), xlim..., label="Bayes(a=$a)", c=3, ls=:dashdot)
6 P4 = plot(p → pvalue_cp(k, n, p), xlim..., label="Clopper-Pearson")
7 plot!(p → pvalue_wilson(k, n, p), xlim..., label="Wilson", c=2, ls=:dash)
8 plot!(p → pvalue_bayes(k, n, p; a), xlim..., label="Bayes(a=$a)", c=3, ls=:dashdot)
9
10 plot(P1, P2, P3, P4; size=(800, 500), plot_title="data: n = $n, k = $k")

```

Out[9]:

data: n = 100, k = 30



3つのP値函数が近似的に一致しており、特にJeffreys事前分布でのBayes版のP値函数は正規分布近似を使って定義されたWilson版のP値函数と非常によく一致している。

In [10]:

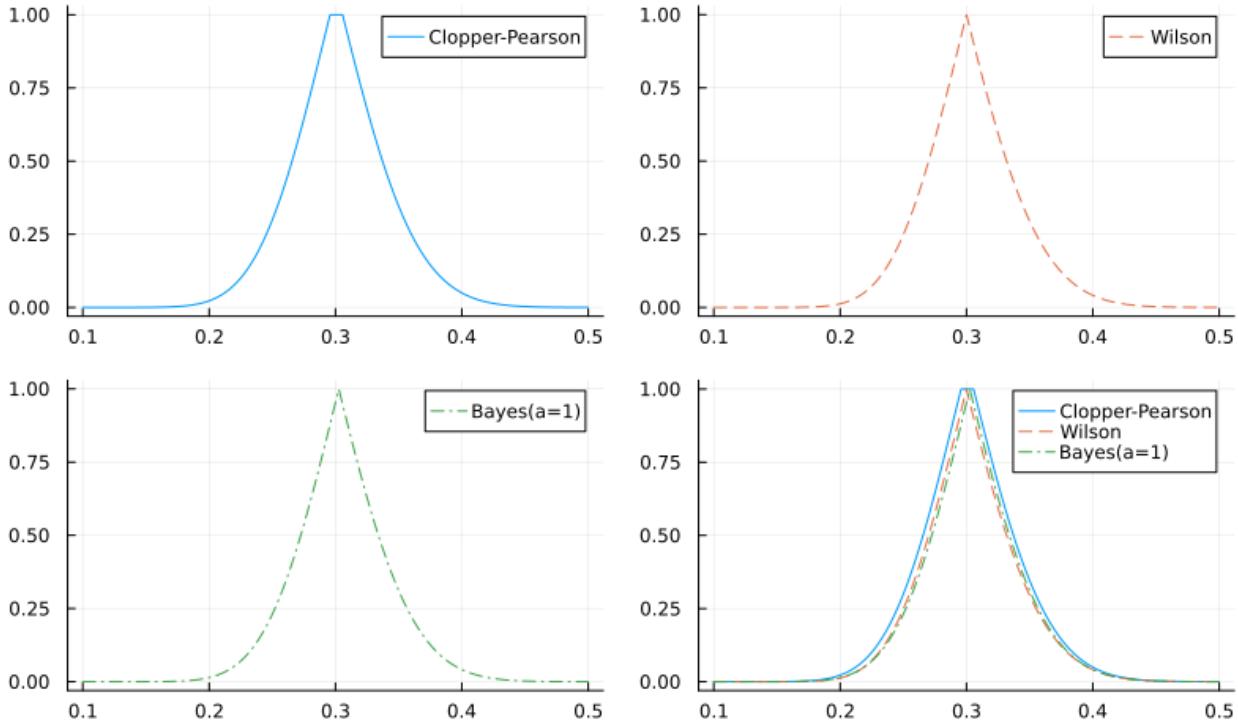
```

1 n, k, a = 100, 30, 1
2 xlim = (0.1, 0.5)
3 P1 = plot(p → pvalue_cp(k, n, p), xlim...; label="Clopper-Pearson")
4 P2 = plot(p → pvalue_wilson(k, n, p), xlim...; label="Wilson", c=2, ls=:dash)
5 P3 = plot(p → pvalue_bayes(k, n, p; a), xlim...; label="Bayes(a=$a)", c=3, ls=:dashdot)
6 P4 = plot(p → pvalue_cp(k, n, p), xlim...; label="Clopper-Pearson")
7 plot!(p → pvalue_wilson(k, n, p), xlim...; label="Wilson", c=2, ls=:dash)
8 plot!(p → pvalue_bayes(k, n, p; a), xlim...; label="Bayes(a=$a)", c=3, ls=:dashdot)
9
10 plot(P1, P2, P3, P4; size=(800, 500), plot_title="data: n = $n, k = $k")

```

Out[10]:

data: n = 100, k = 30



平坦事前分布の場合にも、Bayes版のP値函数はWilson型のP値函数とよく一致している。

以上のグラフを見ても、P値を使う統計学とBayes統計が水と油だと考えることはバカげているように思われる。

1.5 復習: 二項分布とベータ分布の関係の証明

以上を理解できた人は二項分布とベータ分布の累積分布函数の関係を完璧に理解しておくことの価値は大きい感じると思われる。

以下では、再度、二項分布とベータ分布の累積分布函数の関係を証明しておく。

大事なことは何度も証明しておいた方がよい。

1.5.1 両辺を p で微分して確認する方法

証明したい公式

$$1 - \text{cdf}(\text{Binomial}(n, p), k - 1) = \text{cdf}(\text{Beta}(k, n - k + 1), p) \quad (k = 1, 2, \dots, n)$$

の両辺を具体的に書き下すと以下のようになる：

$$\sum_{i=k}^n \binom{n}{i} p^i (1-p)^{n-i} = \frac{\int_0^p t^{k-1} (1-t)^{n-k} dt}{B(k, n - k + 1)} \quad (k = 1, 2, \dots, n). \quad (*)$$

$p = 0$ のとき、 $k \geq 1$ より(*)の左辺は 0 になり、(*)の右辺も 0 になる。

(*)の左辺を p で微分すると、

$$\begin{aligned}
\frac{\partial}{\partial p} ((*) \text{の左辺}) &= \sum_{i \geq k} \frac{n!}{i!(n-i)!} i p^{i-1} (1-p)^{n-i} - \sum_{i \geq k} \frac{n!}{i!(n-i)!} (n-i) p^i (1-p)^{n-i-1} \\
&= \sum_{i \geq k} \frac{n!}{(i-1)!(n-i)!} p^{i-1} (1-p)^{n-i} - \sum_{i \geq k} \frac{n!}{i!(n-i-1)!} p^i (1-p)^{n-i-1} \\
&= \sum_{i \geq k} \frac{n!}{(i-1)!(n-i)!} p^{i-1} (1-p)^{n-i} - \sum_{i \geq k+1} \frac{n!}{(i-1)!(n-i)!} p^{i-1} (1-p)^{n-i} \\
&\quad \frac{n!}{(k-1)!(n-k)!} p^{k-1} (1-p)^{n-k}.
\end{aligned}$$

3つめの等号は2つめの和の項のインデックスの i を $i-1$ で置き換えることによって得られる。

一方, $\Gamma(i+1) = i!$ ($i = 0, 1, 2, \dots$)より,

$$\frac{1}{B(k, n-k+1)} = \frac{\Gamma(n+1)}{\Gamma(k)\Gamma(n-k+1)} = \frac{n!}{(k-1)!(n-k)!}$$

なので, (*)の右辺を p で微分すると,

1.5.2 ベータ分布の一様分布の順序統計量の分布として解釈を使う方法

前節での(*)の証明法では公式(*)が成立する理由はよくわからないので, この節では一様分布の順序統計量としてベータ分布が解釈できることを使った照明を紹介しよう。

T_1, \dots, T_n は 0 から 1 のあいだの n 個の独立な一様乱数であるとし, その中で k 番目に小さなものを $T_{(k)}$ と書く($T_{(k)}$ を一様分布の順序統計量と呼ぶ)。

このとき, $T_{(k)} \leq p$ となることと, T_1, \dots, T_n の中に p 以下のものが k 個以上あることは同値である。各々の T_i が p 以下になる確率は p なので, T_1, \dots, T_n の中に含まれる p 以下のものの個数は二項分布 $\text{Binomial}(n, p)$ に従う。したがって,

$$(T_{(k)} \leq p \text{ となる確率}) = (\text{二項分布 } \text{Binomial}(n, p) \text{ において } k \text{ 以上になる確率}) = \sum_{i \geq k} \binom{n}{i} p^i (1-p)^{n-i}.$$

一方, $t \leq T_{(k)} \leq t + dt$ となる確率は

$$\begin{aligned}
&(t \leq T_{(k)} \leq t + dt \text{ となる確率}) \\
&= (T_{(1)} \leq \dots \leq T_{(k-1)} \leq T_{(k)} \leq t + dt \text{かつ } t \leq T_{(k)} \leq T_{(k+1)} \leq \dots \leq T_{(n)} \text{ となる確率}) \\
&\approx (T_{(1)} \leq \dots \leq T_{(k-1)} \leq t \leq T_{(k)} \leq t + dt \leq T_{(k+1)} \leq \dots \leq T_{(n)} \text{ となる確率}) \\
&= (T_1, \dots, T_n \text{ のうち } k-1 \text{ 個が } t \text{ 以下で, } 1 \text{ 個が } [t, t+dt] \text{ に含まれ, } n-k \text{ 個が } t+dt \text{ 以上の確率}) \\
&= \frac{n!}{(k-1)!1!(n-k)!} t^{k-1} dt (1-t)^{n-k}
\end{aligned}$$

と近似され, 誤差は dt より高次の微小量になる。 $T_{(1)} \leq T_{(2)} \leq \dots \leq T_{(n)}$ が T_i 達を小さな順序に並べたものになっていることおよび, $n!/((k-1)!1!(n-k)!)$ が n 個の T_1, \dots, T_n を $k-1$ 個, 1 個, $n-k$ 個にグループ分けする方法全体の個数(多項係数の特別な場合)になっていることに注意せよ。そして,

$$\frac{n!}{(k-1)!1!(n-k)!} = \frac{\Gamma(n+1)}{\Gamma(k)\Gamma(n-k+1)} = \frac{1}{B(k, n-k+1)}$$

なので, $T_{(k)}$ が従う分布の確率密度函数はベータ分布 $\text{Beta}(k, n-k+1)$ の密度函数

$$p(t) = \frac{t^{k-1} (1-t)^{n-k}}{B(k, n-k+1)}$$

に一致することがわかる。したがって,

$$(T_{(k)} \leq p \text{ となる確率}) = \int_0^p p(t) dt = \frac{\int_0^p t^{k-1} (1-t)^{n-k} dt}{B(k, n-k+1)}.$$

これを上で示した結果と比較することによって次が得られる:

$$\sum_{i=k}^n \binom{n}{i} p^i (1-p)^{n-i} = \frac{\int_0^p t^{k-1} (1-t)^{n-k} dt}{B(k, n-k+1)}.$$

2 P値函数について

P値函数に関する考え方の簡潔な説明については次の論文を参照せよ:

- Valentin Amrhein and Sander Greenland, Discuss practical importance of results based on interval estimates and p-value functions, not only on point estimates and null p-values, Journal of Information Technology, First Published June 3, 2022.

2.1 P値はモデルのパラメータ値とデータの数値の整合性の指標の1つ

我々は、データの数値の生成のされ方(未知の法則)をパラメータ付きの確率分布でモデル化し、それを統計モデルと呼ぶのであつた。

例えば、「 n 回中 k 回当たりが出た」の形のデータの生成のされ方の統計モデルの代表例は二項分布モデル $\text{Binomial}(n, p)$ であり、当たりが出る確率は成功確率パラメータ p でモデル化される。

以下では、データの数値 x の生成のされ方をパラメータ θ を持つ統計モデル $M(\theta)$ でモデル化している状況を考える。

そしてさらに、データの数値 x に関する仮説 $\theta = \theta_0$ のP値 $\text{pvalue}(x|\theta = \theta_0)$ が適切に定義されているという一般的な状況も考える。

P値は、データの数値 x と仮説 $\theta = \theta_0$ の下での統計モデル $M(\theta = \theta_0)$ の整合性の指標の1つである。

ここでの整合性は英語では compatibility (両立性) や consistency (無矛盾性) という意味である。

例えば、二項分布モデルの場合には、「100回中40回当たりが出た」というデータが得られたとき、仮説 $p = 1/2$ のP値 (Clopper-Pearsonの信頼区間を与えるP値) は約 5.7% になる。

その 5.7% というP値は「100回中40回当たりが出た」というデータと「当たりが出る確率は $p = 1/2$ である」という仮説の整合性の指標として使われることになる。

ただし、統計モデルの妥当性に注意を払う必要がある。

P値の定義の仕方によって、その値は微妙に異なった値になる。

そういう細かな違いに結果が影響されないような頑健な議論を目指すべきである。

```
In [11]: 1 @show k, n, p = 40, 100, 1/2
2 @show pvalue_cp(k, n, p)
3 @show pvalue_sterne(k, n, p)
4 @show pvalue_wilson(k, n, p)
5 @show pvalue_wald(k, n, p);
```

```
(k, n, p) = (40, 100, 1 / 2) = (40, 100, 0.5)
pvalue_cp(k, n, p) = 0.05688793364098078
pvalue_sterne(k, n, p) = 0.05688793364098054
pvalue_wilson(k, n, p) = 0.04550026389635841
pvalue_wald(k, n, p) = 0.041226833337163676
```

2.2 P値は信頼区間を考えたいすべてのパラメータ値について定義されている

二項分布モデルの場合には仮説 $p = 1/2$ に限らず、0から1のあいだの任意の数値 p_0 に関する仮説 $p = p_0$ についてP値が定義されている。

それによって、「100回中40回当たりが出た」というデータが得られたとき、仮説 $p = p_0$ における p_0 を動かしながらP値を計算することによって、 $p = 1/2$ の場合以外についても仮説 $p = p_0$ と「100回中40回当たりが出た」というデータの整合性達がどうなっているかを知ることができる。

例えば、仮説 $p = 0.3$ とデータの数値の整合性を知ることができる。

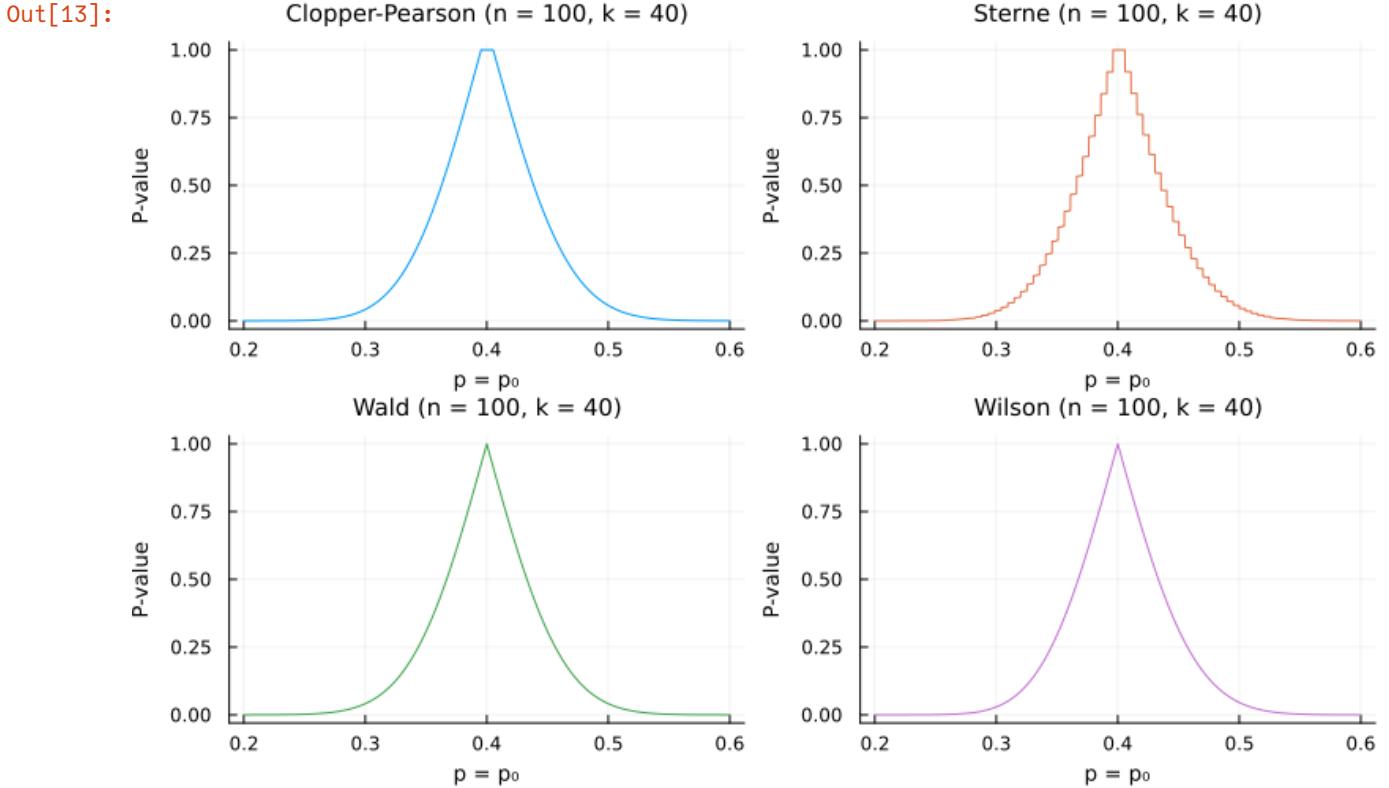
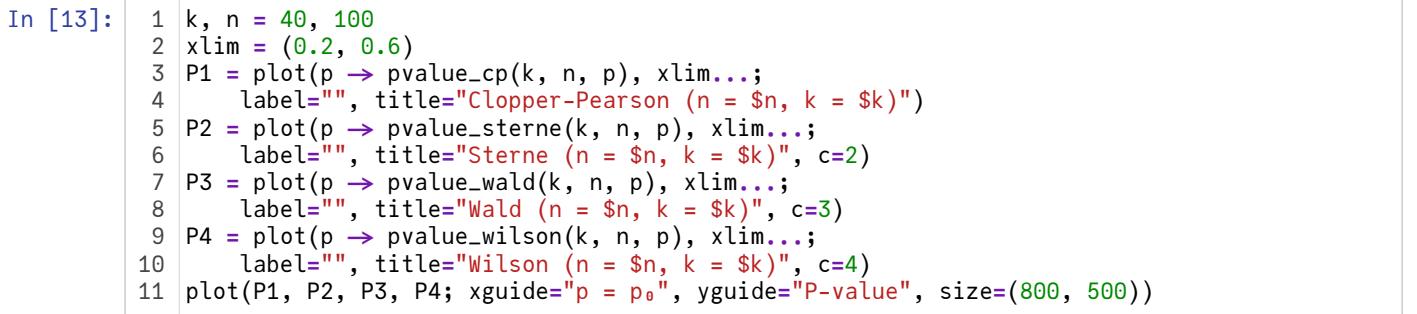
```
In [12]: 1 @show k, n, p = 40, 100, 0.3
2 @show pvalue_cp(k, n, p)
3 @show pvalue_sterne(k, n, p)
4 @show pvalue_wilson(k, n, p)
5 @show pvalue_wald(k, n, p);
```

```
(k, n, p) = (40, 100, 0.3) = (40, 100, 0.3)
pvalue_cp(k, n, p) = 0.04197715200784929
pvalue_sterne(k, n, p) = 0.03745142924579382
pvalue_wilson(k, n, p) = 0.029096331741252188
pvalue_wald(k, n, p) = 0.041226833337163676
```

一般に、パラメータ値 θ_0 に対して、データ x に関する仮説 $\theta = \theta_0$ のP値を対応させる函数 $\theta_0 \mapsto \text{pvalue}(x|\theta = \theta_0)$ を **P値函数** (P-value function) と呼ぶ。

P値函数は、統計モデルについて、データの数値とパラメータ値のあいだの整合性(相性)の様子全体の情報を持っている重要な函数である。

以下では、二項分布モデルにおける4種のP値関数のグラフをプロットしてみよう。



近似的にはどれもほぼ同じ形をしている。

2.3 分割表の場合のP値関数

「Aでは $m = a + b$ 回中 a 回当たりが出で、Bでは $n = c + d$ 回中 c 回当たりが出た」の形式のデータを

	当たり	はずれ	合計
A	a	b	m
B	c	d	n

のように 2×2 の分割表で表すのであった。

このようなデータの生成のされ方の統計モデルとして、2つの二項分布モデル $\text{Binomial}(m, p) \times \text{Binomial}(n, q)$ を採用しよう。

このモデル内で、 a の値は二項分布 $\text{Binomial}(m, p)$ に従ってランダムに決まり、 c の値はそれとは独立に二項分布 $\text{Binomial}(n, q)$ に従ってランダムに決まっていると考える。 $(m = a + b$ と $n = c + d$ は固定されているので、 a, c から b, d が自動的に決まる。)

応用時に重要なのは、 p と q の違いである。

入門的な多くの教科書では非常に残念なことに、仮説 $p = q$ のP値の定義しか書かれておらず、 p と q の違いの大きさを表す指標の信頼区間の定義が書かれていません。

これが前節で、P値は信頼区間を考えたいすべてのパラメータ値について定義されていると強調したくなった理由である。

しかし、我々はすでに

- 「検定と信頼区間: 比率の比較」のノート

(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/11%20Hypothesis%20testing%20and%20confidence%20Two%20proportions.ipynb>)

で p と q の違いを表す3つの指標についてP値と信頼区間を定義している:

- オッズ比パラメータ $OR = \frac{p/(1-p)}{q/(1-q)} = \frac{p(1-q)}{(1-p)q}$,
- リスク比パラメータ $RR = \frac{p}{q}$,
- リスク差パラメータ $RD = p - q$.

これは、このノート群が入門的な教科書の多くと一線を画する点である。

この点に関しては、Rothmanさん達の有名な疫学の教科書

- Rothman, Lash, and Greenland, Modern Epidemiology [[Googleで検索 \(https://www.google.com/search?q=Rothman+Lash+Greenland+Modern+Epidemiology\)](https://www.google.com/search?q=Rothman+Lash+Greenland+Modern+Epidemiology)]

が詳しい。講義動画

- 佐藤俊哉、臨床研究者のための生物統計学「回帰モデルと傾向スコア」2019年2月21日 [[YouTube \(https://youtu.be/cOHN444kBlo\)](https://youtu.be/cOHN444kBlo)]

も参照せよ。8:30以降の「変換しない場合」がリスク差パラメータを p と q の違いの指標として採用した場合に対応しており、10:30以降の「対数変換の場合」がリスク比パラメータを p と q の違いの指標として採用した場合に対応しており、12:00以降の「ロジット変換の場合」がオッズ比パラメータ(正確にはその対数)を p と q の違いの指標として採用した場合に対応している。「ロジット変換の場合」はロジスティック回帰の話になっている。ただし、複数の分割表のデータを扱うより複雑な場合を扱っている。

注意: 実際にはオッズ比パラメータの代わりに対数オッズ比パラメータ $\log OR$ を扱うことが多い。その場合はロジスティック・モデルの特別な場合になっている。**注意終**

2.4 P値函数から信頼区間が定義される

P値が信頼区間を考えたいすべてのパラメータ値について定義されていることの利点は、言うまでもないことだが、信頼区間を定義できることである。

一般に、データ x の生成のされ方に関する統計モデル $M(\theta)$ に関するP値函数 $pvalue(x|\theta = \theta_0)$ が与えられているとき、パラメータ θ に関する信頼度 $1 - \alpha$ の信頼区間 ($100(1 - \alpha)\%$ 信頼区間, confidence interval)は次のように定義される:

$$\text{confint}^\theta(x|\alpha) = \{ \theta_0 \mid pvalue(x|\theta = \theta_0) \geq \alpha \}.$$

すなわち、データの数値 x について仮説 $\theta = \theta_0$ のP値が α 以上になるような θ_0 全体の集合が信頼区間になる。

ここで使われている閾値 α は**有意水準** (significance level)と呼ばれる。

$1 - \alpha$ は**信頼度** (信頼水準, confidence level)と呼ばれる。

コンピュータで計算するときには、 $pvalue(x|\theta = \theta_0) = \alpha$ を満たす θ_0 の値を2分法やNewton法などで計算すれば、一般的な場合にも信頼区間を概ね計算できる。

しかし、それだとバグも発生し易くなるし、計算効率も悪くなりがちなので、信頼区間を直接できるシンプルな公式がある場合にはそちらを使うことになる。

2.5 2×2の分割表のP値と信頼区間とP値函数の例

以下では2×2の分割表のデータの数値

	当たり	はずれ
A	$a = 30$	$b = 70$
B	$c = 20$	$d = 80$

に関するP値と信頼区間の例を示そう。

簡単のためオッズ比パラメータ OR を扱う場合(実際には対数オッズ比パラメータ $\log OR$ を扱う場合)のWald型のP値函数のみを扱う。

$a + b = m, c + d = n$ が十分に大きければ、仮説 $OR = \frac{p/(1-p)}{q/(1-q)} = \omega$ の統計モデル内で、モデル内での仮想的データ a, b, c, d の数値の対数オッズ比

$$\log \widehat{OR} = \log \frac{ad}{bc}$$

が次のように近似的に正規分布に従うことを示せる:

$$\log \widehat{\text{OR}} \sim \text{Normal}\left(\log \omega, \widehat{\text{SE}}_{\log \widehat{\text{OR}}}\right), \text{ approximately.}$$

ここで,

$$\widehat{\text{SE}}_{\log \widehat{\text{OR}}} = \sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}$$

この近似については

- 「検定と信頼区間: 比率の比較」のノート

(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/11%20Hypothesis%20testing%20and%20confidence%20%20Two%20proportions.ipynb>)

の「Wald版のオッズ比に関するP値と信頼区間の定義」に関する節を参照せよ.

これによって、以下のように仮説 $\text{OR} = \omega$ のP値と OR の信頼区間が定義できる:

$$\begin{aligned} \text{pvalue}_{\text{Wald}}(a, b, c, d | \text{OR} = \omega) &= 2 \left(1 - \text{cdf} \left(\text{Normal}(0, 1), \frac{|\log \widehat{\text{OR}} - \log \omega|}{\widehat{\text{SE}}_{\log \widehat{\text{OR}}}} \right) \right), \\ \text{confint}_{\text{Wald}}^{\text{OR}}(a, b, c, d | \alpha) &= \left[\exp \left(-z_{\alpha/2} \widehat{\text{SE}}_{\log \widehat{\text{OR}}} \right) \widehat{\text{OR}}, \exp \left(z_{\alpha/2} \widehat{\text{SE}}_{\log \widehat{\text{OR}}} \right) \widehat{\text{OR}} \right]. \end{aligned}$$

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

注意: $\text{pvalue}_{\text{Wald}}(a, b, c, d | \text{OR} = \omega)$ の定義の右辺は、標準正規分布内で

$$\frac{\log \widehat{\text{OR}} - \log \omega}{\widehat{\text{SE}}_{\log \widehat{\text{OR}}}}$$

の絶対値以上の(データの数値以上極端な)数値が生成される確率の2倍になっている.

```
In [14]: # 上の定義通りのP値函数と信頼区間函数の実装
1 oddsratiohat(a, b, c, d) = safediv(a*d, b*c)
2 stderrhat_logoddsratiohat(a, b, c, d) = sqrt(1/a + 1/b + 1/c + 1/d)
3
4 function pvalue_or_wald(a, b, c, d; ω=1)
5     logORhat = log(oddsratiohat(a, b, c, d))
6     SEhat_logORhat = stderrhat_logoddsratiohat(a, b, c, d)
7     2*ccdf(Normal(0, 1), safediv(abs(logORhat - log(ω)), SEhat_logORhat))
8 end
9
10 function confint_or_wald(a, b, c, d; α=0.05)
11     z = quantile(Normal(), 1-α/2)
12     ORhat = oddsratiohat(a, b, c, d)
13     SEhat_logORhat = stderrhat_logoddsratiohat(a, b, c, d)
14     [safemul(exp(-z*SEhat_logORhat), ORhat), safemul(exp(z*SEhat_logORhat), ORhat)]
15 end
16
17
```

Out[14]: confint_or_wald (generic function with 1 method)

In [15]:

```

1 # グラフのプロット用の函数
2 # この手の函数は計算用の函数よりも複雑になりがち
3
4 function plot_pvalue_function_of_or_wald();
5     a=30, b=70, c=20, d=80, ω = 1.0, α = 0.05,
6     xlim = confint_or_wald(a, b, c, d; α=1e-3)
7 )
8 @show α
9 @show ω
10 @show z = quantile(Normal(0, 1), 1 - α/2)
11 println()
12 @show ORhat = oddsratiohat(a, b, c, d)
13 @show exp(z * stderrhat_logoddsratiohat(a, b, c, d))
14 println()
15 @show P_value = pvalue_or_wald(a, b, c, d; ω)
16 @show CI = confint_or_wald(a, b, c, d; α)
17
18 plot(ω → pvalue_or_wald(a, b, c, d; ω), xlim...;
19       label="P-value function (Wald)", c=:black)
20 vline!([ORhat]; label="ORhat = $(round(ORhat; digits=5))", ls=:dash, c=2)
21 plot!(CI, fill(α, 2); label="$(100(1-α))% confidence interval", lw=2, c=3)
22 scatter!([ω], [α]; label="significance level α = $(100α)%", c=3, msc=:auto)
23 vline!([ω]; label="", c=:black, lw=0.5)
24 scatter!([ω], [P_value]; label="P-value of OR = $ω", c=:red, msc=:auto)
25 plot!(xguide="OR = ω (log scale)", yguide="P-value")
26 plot!(xscale=:log, xtick=logtick(; xlim), ytick=0:0.05:1)
27 title!("a, b, c, d = $a, $b, $c, $d")
28 plot!(size=(720, 350), bottommargin=4Plots.mm)
29 end

```

Out[15]: plot_pvalue_function_of_or_wald (generic function with 1 method)

In [16]:

```

1 a, b, c, d = 30, 70, 20, 80
2 graph_of_pvalue_function =
3     plot_pvalue_function_of_or_wald(; a, b, c, d, ω = 1.0, α = 0.05)

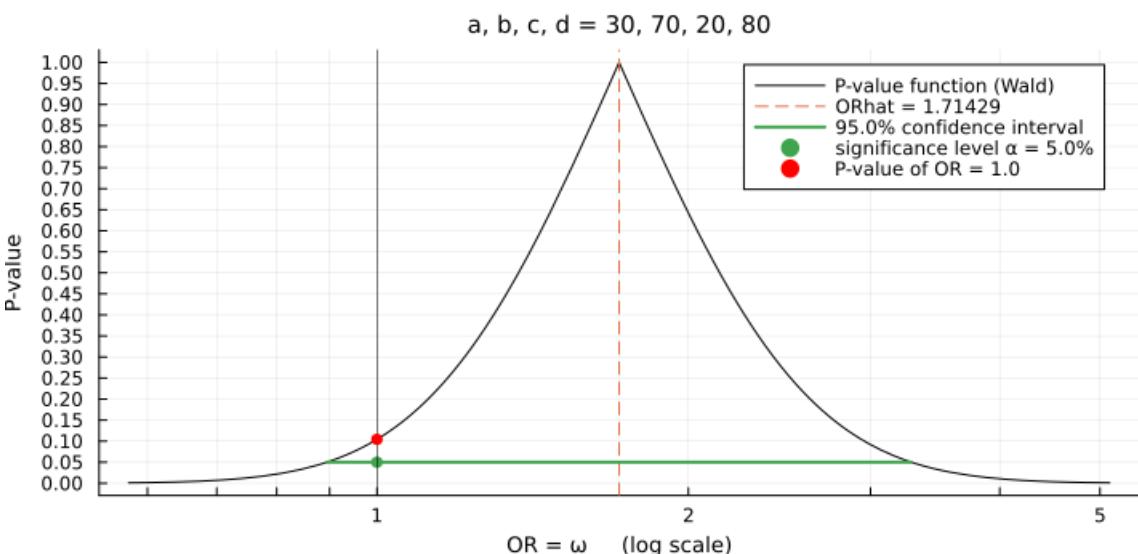
α = 0.05
ω = 1.0
z = quantile(Normal(0, 1), 1 - α / 2) = 1.9599639845400576

ORhat = oddsratiohat(a, b, c, d) = 1.7142857142857142
exp(z * stderrhat_logoddsratiohat(a, b, c, d)) = 1.9163037024703367

P_value = pvalue_or_wald(a, b, c, d; ω) = 0.10432099005636014
CI = confint_or_wald(a, b, c, d; α) = [0.8945793467266185, 3.28509206137772]

```

Out[16]:



仮説 $OR = 1$ すなわち「AとBで当たりが出る確率は同じ」という仮説のP値は 10.4% 程度になっている。

これが意味するところは以下の通り。データの数値

	当たり	はずれ
A	$a = 30$	$b = 70$
B	$c = 20$	$d = 80$

を見ると、Aの方がBよりも当たりが出る確率が高そうに見えるが、統計モデル内ではAとBで当たりが出る確率が同じであった
さらに、上のグラフを見ると、仮説 $OR = 4$ のP値は1%程度に見えるが、実際に計算してもその程度の値になる。

In [17]: 1 pval4 = pvalue_or_wald(a, b, c, d; ω=4)

Out[17]: 0.010670206411228012

WolframAlphaでも仮説 $OR = 4$ のP値を計算してみよう。

$\widehat{SE}_{\log OR} \rightarrow \sqrt{1/a + 1/b + 1/c + 1/d}$ where $a=30.0, b=70, c=20, d=80 \rightarrow$ [実行

(<https://www.wolframalpha.com/input?i=sqrt%281%2Fa+1%2B+1%2Fb+1%2Fc+1%2Fd%29+where+a%3D30.0%2C+b%3D70%2C+c%3D20%2C+d%3D80>)

$\rightarrow 0.331842$

$\log \widehat{OR} - \log \omega \rightarrow \log((a/d)/(b/c)) - \log(4)$ where $a=30.0, b=70, c=20, d=80 \rightarrow$ [実行

(<https://www.wolframalpha.com/input?i=log%28a+d%29%2F%28b+c%29%29+->

$+log%284%29+where+a%3D30.0%2C+b%3D70%2C+c%3D20%2C+d%3D80)] \rightarrow -0.847298$

$(\log \widehat{OR} - \log \omega) / \widehat{SE}_{\log OR} \rightarrow 0.847298 / 0.331842 \rightarrow$ [実行 (<https://www.wolframalpha.com/input?i=0.847298%2F0.331842>)]

$\rightarrow 2.55332$

P値 $\rightarrow 2(1 - cdf(NormalDistribution(0,1), 2.55332)) \rightarrow$ [実行 (<https://www.wolframalpha.com/input?i=2%281+-cdf%28NormalDistribution%280%2C1%29%29%29%29>)]

$\rightarrow 0.0106701$

In [18]: 1 # WolframAlphaで求めたP値のJuliaで求めたP値に対する相対誤差が小さいことの確認

2 0.0106701/pval4 - 1

Out[18]: -9.972743160724384e-6

WolframAlphaでも OR の 95% 信頼区間を求めてみよう。

$z_{\alpha/2} \rightarrow quantile(Normal(0,1), 0.975) \rightarrow$ [実行 (<https://www.wolframalpha.com/input?i=quantile%28Normal%280%2C1%29%29%29>)]

$\rightarrow 1.95996$

信頼区間 $\rightarrow \{\exp(-1.95996 * 0.331842) (a/d)/(b/c), \exp(1.95996 * 0.331842) (a/d)/(b/c)\}$ where

$a=30.0, b=70, c=20, d=80 \rightarrow$ [実行 (https://www.wolframalpha.com/input?i=%7Bexp%28-1.95996*0.331842%29%28a+d%29%2F%28b+c%29%2C+exp%281.95996*0.331842%29%28a+d%29%2F%28b+c%29%29)]

$\rightarrow \{0.89458, 3.28509\}$

In [19]: 1 # WolframAlphaで求めた信頼区間のJuliaで求めた信頼区間にに対する相対誤差が小さいことの確認

2 [0.89458, 3.28509] ./ confint_or_wald(a, b, c, d; α=0.05) .- 1

Out[19]: 2-element Vector{Float64}:

7.302576165990615e-7

-6.274946581230623e-7

2.6 2×2の分割表での検定ではどれを使うべきか

以上においては計算の仕方がシンプルだという理由でWald型のP値函数と信頼区間を扱ったが、標本サイズが小さいときに誤差が大きくなるという欠点がある。他にも以下の選択肢がある：

(1) Pearsonの χ^2 検定 (ただし扱う仮説を $OR = \omega$ と $RR = \rho$ の場合に一般化しておくこと、連続性補正はかけない)

(2) Fisher検定 (ただし扱う仮説を $OR = \omega$ の場合に一般化しておくこと)

(3) p と q の違いの指標としてそれらの差 $p - q$ を扱う場合には Zou-Donner 2004 の方法が使える。[\(Julia言語による実装例](https://github.com/genkuroki/public/blob/main/0033/probability%20of%20alpha%20error%20of%20Zou-Donner.ipynb) (<https://github.com/genkuroki/public/blob/main/0033/probability%20of%20alpha%20error%20of%20Zou-Donner.ipynb>))

それぞれ長所と短所があるので目的に合わせて適当に使い分ければよい。標本サイズが十分に大きい場合にはWald型のP値函数と信頼区間も十分な精度を持ち、問題なく使える。

注意：よく使われているのは、Pearsonの χ^2 検定とFisher検定である。標本サイズが小さい場合にはFisher検定を使えと解説されている場合もあるようだが、実際に色々計算するとそれは誤りだと分かる。実際には標本サイズが小さなとき、Fisher検定には過剰にP値が大きくなり過ぎるという欠点がある。しかし、その欠点は「第一種の過誤の確率が常に有意水準以下である」というFisher検定の優れた性質の裏返しなので、どの条件を重視するかによって、(Yatesの連続性補正をかけない)Pearsonの χ^2 検定と使い分けるとよいと思われる。次の解説も参考になる：

- 朝倉こう子, 濱崎俊光, 連載 第3回 医学データの統計解析の基本 2つの割合の比較 [PDF] (https://www.jstage.jst.go.jp/article/dds/30/3/30_235/_pdf)

第一種の過誤の確率を確実に名目有意水準以下にしたければFisher検定を使う方がよいし, 平均的により高い検出力を得たい

2.7 P値函数と最尤法の関係

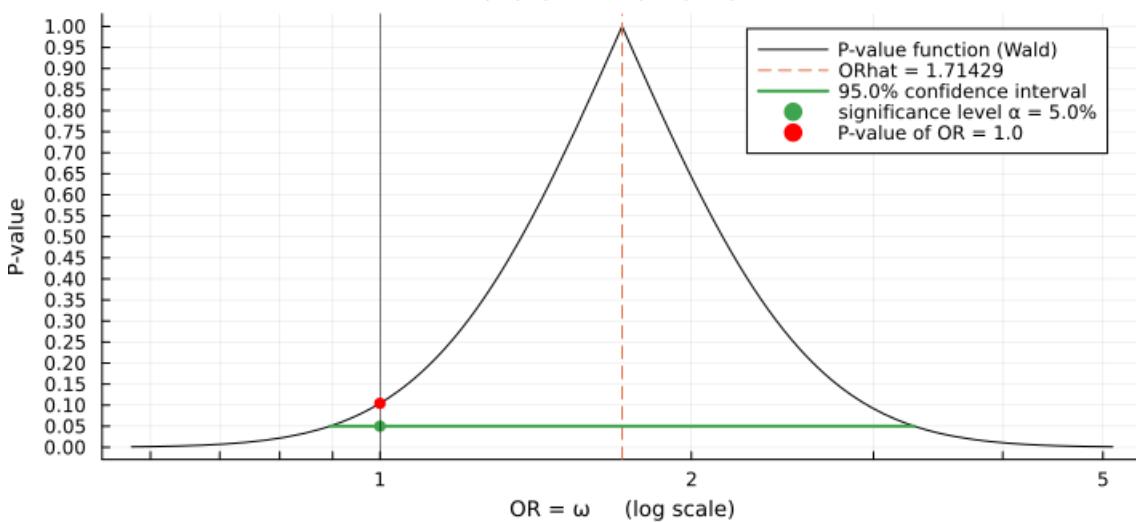
2×2の分割表のデータの数値

	当たり	はずれ
A	$a = 30$	$b = 70$
B	$c = 20$	$d = 80$

に関するP値函数などは1つのグラフの中に以下のようにプロットされるのであった.

In [20]: 1 graph_of_pvalue_function

Out[20]:



このグラフにおけるデータの数値のオッズ比

$$\widehat{OR} = \frac{ad}{bc} = \frac{30 \cdot 80}{70 \cdot 20} = \frac{12}{7} \approx 1.7142857142857142$$

の値でP値函数が最大値の 1 になっていることに注意せよ. P値函数の定義

$$pvalue_{Wald}(a, b, c, d | OR = \omega) = 2 \left(1 - \text{cdf} \left(\text{Normal}(0, 1), \frac{|\log \widehat{OR} - \log \omega|}{\widehat{\text{SE}}_{\log \widehat{OR}}} \right) \right)$$

を見れば, $\omega = \widehat{OR}$ のとき最大値 1 になることはすぐにわかる.

実はこのデータの数値のオッズ比 $OR = (ad)/(bc)$ は2つの二項分布モデルにおける最尤法(さいゆうほう)の解

$$\hat{p} = \frac{a}{a+b}, \quad \hat{q} = \frac{c}{c+d} \quad \left(\text{このとき } 1 - \hat{p} = \frac{b}{a+b}, \quad 1 - \hat{q} = \frac{d}{c+d} \right)$$

のオッズ比になっている:

$$\frac{\hat{p}(1 - \hat{q})}{(1 - \hat{p})\hat{q}} = \frac{ad}{bc} = \widehat{OR}.$$

この場合に限らず, 多くの場合にP値函数が最大になるパラメータ値は最尤法の解に対応する値になっている.

少なくとも, 近似的にはP値函数を最大化するパラメータ値は最尤法の解に対応する値だと考えてよい.

データの数値 x に関する仮説 $\theta = \theta_0$ のP値は, 統計モデルを前提にしたときの仮説 $\theta = \theta_0$ とデータの数値 x の整合性の指標の1つであった.

ゆえに, 以上で述べたことは, 最尤法の解に対応するパラメータ値はデータの数値との整合性が最も高いパラメータ値であるとみなせる.

最尤法による点推定にはこのような解釈がある.

注意・警告: 統計学を実践的に使う場合には、最尤法による点推定の結果のみを報告せずに、信頼区間などの区間推定の結果も報告する習慣になっている。点推定の値は単に統計モデルの前提の下でデータの数値と最も整合性が高いパラメータ値を計算したにすぎず、確率的な揺らぎが考慮されていない。確率的揺らぎの影響に配慮した区間推定の結果も報告しないと、点推定の値がどこまで正確な推定値だと考えられるかが全くわからなくなってしまう。

注意: 尤度(ゆうど, likelihood)については

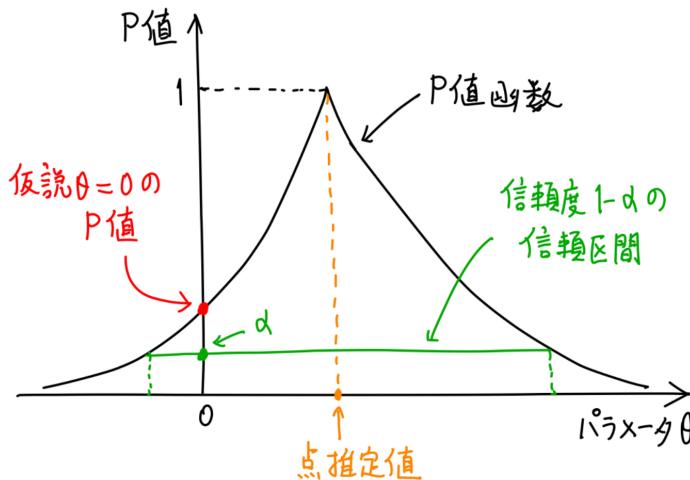
- 「条件付き確率分布、尤度、推定、記述統計」に関するノート

(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/06%20Conditional%20distribution%2C%20likelihood%2C>)

における「尤度(ゆうど)と推定」の節を参照せよ。尤度の定義は **統計モデル内でデータと同じ数値が生成される確率もしくは確率の密度** であり、モデルのデータの数値への適合度の指標としては使えるが、「もっともらしさ」の指標としては不適切で

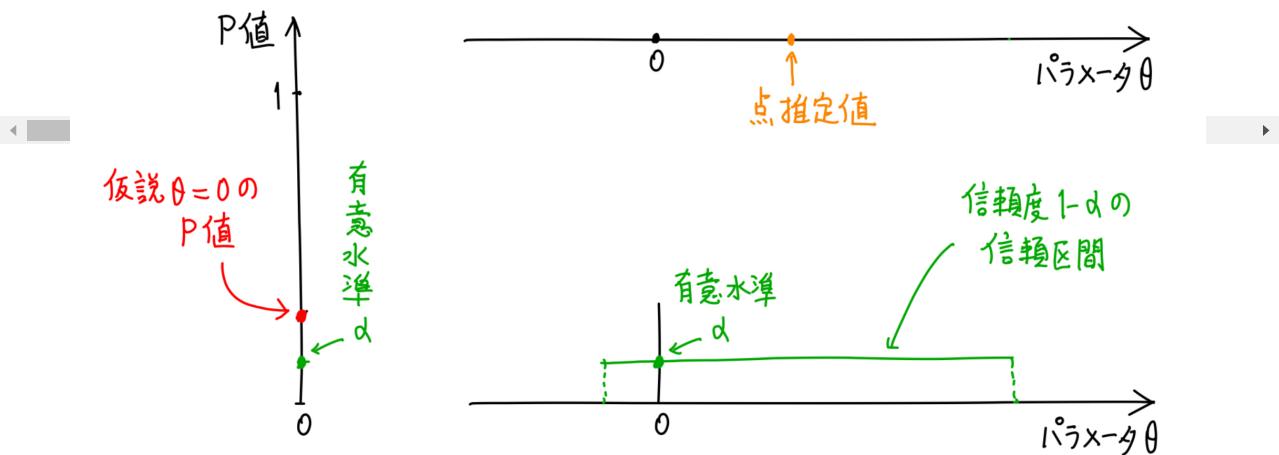
2.8 以上のまとめの図

以下の図のように、P値函数全体の形を見れば多くのことを一目で確認できる。



仮説 $\theta = 0$ の P 値のみ、点推定値のみ、信頼区間のみを P 値函数のグラフ抜きにばらばらに描くと以下のようになり、全体の関係がどうなっているかがひどく分かり難くなる。

仮説 $\theta = 0$ の P 値のみ、点推定値のみ、信頼区間のみをばらばらに描くとこうなる、



注意: Bayes統計における事後分布も以上で説明したP値函数とほぼ同じ使い方をできる。P値函数について十分に理解しておけば、Bayes統計について学習するときにも役に立つだろう。

2.9 おまけ: Bayes統計の方法を使った場合

このノート群の中ではできるだけBayes統計の方法には触れないように気を使ったが、何も触れないわけには行かなかった。現代においては、P値函数を使う方法だけではなく、Bayes統計の方法も「統計学の道具箱」の中に入れておいた方がよいと思われる。

2つの二項分布モデル $\text{Binomial}(m, p) \times \text{Binomial}(n, q)$ の事前分布として、2つのJeffreya事前分布 $\text{Beta}(1/2, 1/2) \times \text{Beta}(1/2, 1/2)$ を採用する。

このとき、 2×2 の分割表のデータの数値

	当たり	はずれ
A	$a = 30$	$b = 70$
B	$c = 20$	$d = 80$

から得られる事後分布は、 $\text{Beta}(30 + 1/2, 70 + 1/2) \times \text{Beta}(20 + 1/2, 80 + 1/2)$ になる。

```
In [21]: 1 a, b, c, d = 30, 70, 20, 80
          2 posterior = product_distribution([Beta(a+1/2, b+1/2), Beta(c+1/2, d+1/2)])
```

Out[21]: $\text{Product}\{\text{Continuous}, \text{Beta}\{\text{Float64}\}, \text{Vector}\{\text{Beta}\{\text{Float64}\}\}\}(\nu = \text{Beta}\{\text{Float64}\}[\text{Beta}\{\text{Float64}\}(\alpha=30.5, \beta=70.5), \text{Beta}\{\text{Float64}\}(\alpha=20.5, \beta=80.5)])$

事後分布のサイズ100万のサンプルを作る。

```
In [22]: 1 sample_of_posterior = rand(posterior, 10^6)
```

Out[22]: $2 \times 1000000 \text{ Matrix}\{\text{Float64}\}:$
$$\begin{matrix} 0.336198 & 0.38694 & 0.30671 & 0.342529 & 0.253606 & \dots & 0.292857 & 0.304909 & 0.408855 & 0.451934 \\ 0.156355 & 0.181258 & 0.195011 & 0.154641 & 0.155184 & & 0.228326 & 0.236909 & 0.216392 & 0.235583 \end{matrix}$$

事後分布でのオッズ比のサンプルを構成する。

```
In [23]: 1 @views p, q = sample_of_posterior[1,:], sample_of_posterior[2,:]
          2 sample_of_posterior_odds_ratio = @. (p*(1-q))/((1-p)*q)
          3 sample_of_posterior_odds_ratio'
```

Out[23]: $1 \times 1000000 \text{ adjoint}(\text{Vector}\{\text{Float64}\})$ with eltype $\text{Float64}:$
$$\begin{matrix} 2.73278 & 2.85094 & 1.82617 & 2.84799 & 1.84972 & \dots & 1.4091 & 1.39968 & 1.41294 & 2.50456 & 2.67564 \end{matrix}$$

オッズ比の点推定値として、事後分布での中央値を採用しよう。

```
In [24]: 1 ORhat_posterior_median = median(sample_of_posterior_odds_ratio)
```

Out[24]: 1.7098834404792975

```
In [25]: 1 ORhat = oddsratiohat(a, b, c, d) # 最尤推定値
```

Out[25]: 1.7142857142857142

最尤推定値の ORhat との違いは小さい。

```
In [26]: 1 ORhat_posterior_median / ORhat - 1
```

Out[26]: -0.0025679930537431117

オッズ比の区間推定として、事後分布での 2.5% と 97.5% 分位点の間を採用しよう。

これを 95% 信用区間(確信区間, credible interval)と呼ぶ。

```
In [27]: 1 credible_interval = quantile.(Ref(sample_of_posterior_odds_ratio), [0.025, 0.975])
          2 print(credible_interval)
```

[0.8994525626226166, 3.3027161703389094]

```
In [28]: 1 confidence_interval = confint_or_wald(a, b, c, d; α=0.05)
          2 print(confidence_interval)
```

[0.8945793467266185, 3.28509206137772]

信頼区間(confidence interval)と信用区間(credible interval)の違いは小さい。

```
In [29]: 1 credible_interval ./ confidence_interval .- 1 ▷ print
```

[0.005447494304255729, 0.0053648752095545316]

オッズ比パラメータが 1 という仮説のP値のBayes的代替物として、事後分布で 1 以下になる確率と 1 以上になる確率の小さい方の2倍(= 1 の小さい方)を採用する。(これはClopper-Pearson型のP値函数の定義の類似になっている。)

```
In [30]: 1 ecdf_OR = ecdf(sample_of_posterior_odds_ratio)
2 pvalue_or_bayesian = ω → min(1, 2ecdf_OR(ω), 2(1 - ecdf_OR(ω)))
3 P_value_Bayesian = pvalue_or_bayesian(1)
```

Out[30]: 0.102108

```
In [31]: 1 Pvalue_Wald = pvalue_or_wald(a, b, c, d; ω=1)
```

Out[31]: 0.10432099005636014

通常のP値とそのBayes的類似の違いは小さい。

```
In [32]: 1 P_value_Bayesian/Pvalue_Wald - 1
```

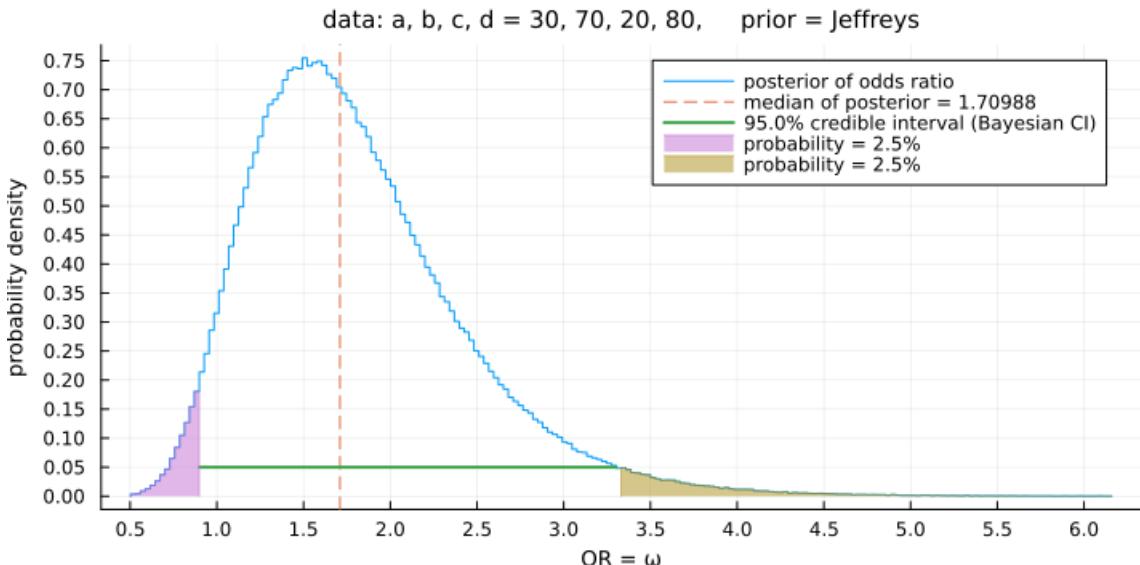
Out[32]: -0.021213276974888284

以上をまとめてプロットしてみよう。

```
In [33]: 1 ORs = sample_of_posterior_odds_ratio
2 CI = credible_interval
3 α = 0.05
4
5 xlim = quantile.(Ref(ORs), (1e-4, 1-1e-4))
6 bin = range(xlim..., 201)
7 h = Plots._make_hist((ORs,), bin, normed=true)
8 y = h.weights
9
10 plot(bin, y; seriestype=:stepbins)
11 stephist(ORs; norm=true, bin,
12         label="posterior of odds ratio")
13 vline!([ORhat_posterior_median];
14        label="median of posterior = $(round(ORhat_posterior_median; digits=5))", ls=:dash, c=2)
15 plot!(xguide="OR = ω", yguide="probability density")
16 plot!(xtick=0:0.5:10, ytick=0:0.05:1)
17 title!("data: a, b, c, d = $a, $b, $c, $d, prior = Jeffreys")
18 plot!(size=(720, 350))
19
20 imin = findlast(i → ecdf_OR(bin[i]) < 0.025, eachindex(bin))
21 imax = findlast(i → ecdf_OR(bin[i]) > 0.975, reverse(eachindex(bin)))
22 @show ecdf_OR(bin[imin]), ecdf_OR(bin[imax])
23
24 plot!(CI, fill(α, 2); label="$(100(1-α))% credible interval (Bayesian CI)", lw=2, c=3)
25 plot!(bin[begin:imin], y[begin:imin-1]; seriestype=:stepbins,
26       label="probability = 2.5%", c=4, fillrange=0, alpha=0.5)
27 plot!(bin[imax:end], y[imax:end]; seriestype=:stepbins,
28       label="probability = 2.5%", c=5, fillrange=0, alpha=0.5)
```

(ecdf_OR(bin[imin]), ecdf_OR(bin[imax])) = (0.024796, 0.976353)

Out[33]:



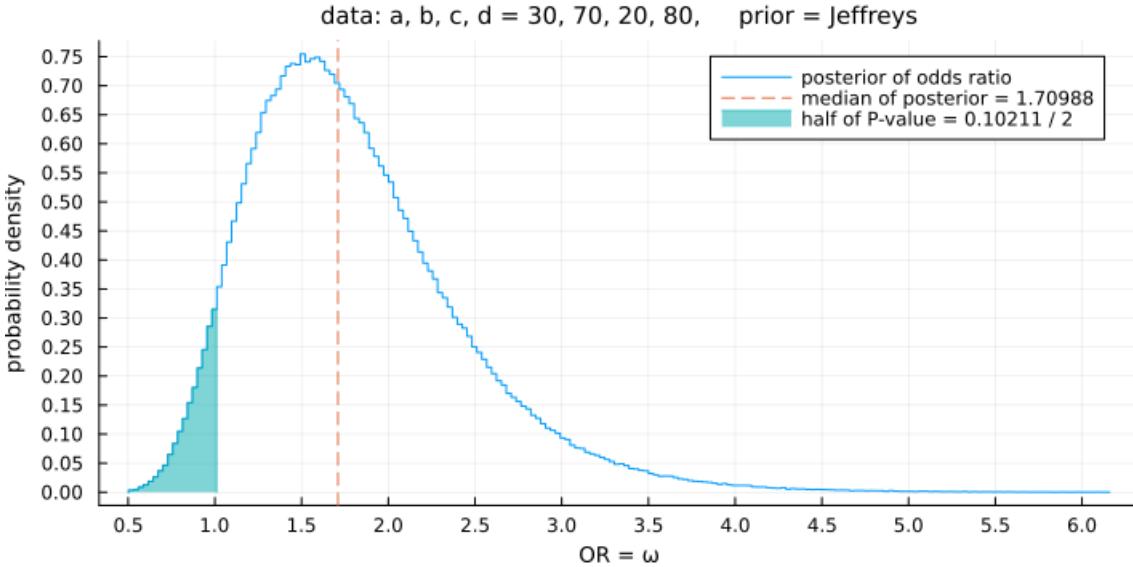
In [34]:

```

1 plot(bin, y; seriestype=:stepbins)
2 stephist(ORs; norm=true, bin,
3         label="posterior of odds ratio")
4 vline!([ORhat_posterior_median];
5        label="median of posterior = $(round(ORhat_posterior_median; digits=5))", ls=:dash, c=2)
6 plot!(xguide="OR = ω", yguide="probability density")
7 plot!(xtick=0:0.5:10, ytick=0:0.05:1)
8 title!("data: a, b, c, d = $a, $b, $c, $d, prior = Jeffreys")
9 plot!(size=(720, 350))
10
11 inull = findlast(i → bin[i] < 1, eachindex(bin)) + 1
12
13 plot!(bin[begin:inull], y[begin:inull-1]; seriestype=:stepbins,
14         label="half of P-value = $(round(P_value_Bayesian; digits=5)) / 2", c=6, fillrange=0, a=1)

```

Out[34]:



通常のP値函数(Wald)とBayes的なP値函数の類似物のグラフを同時に描くとほぼぴったり重なる。

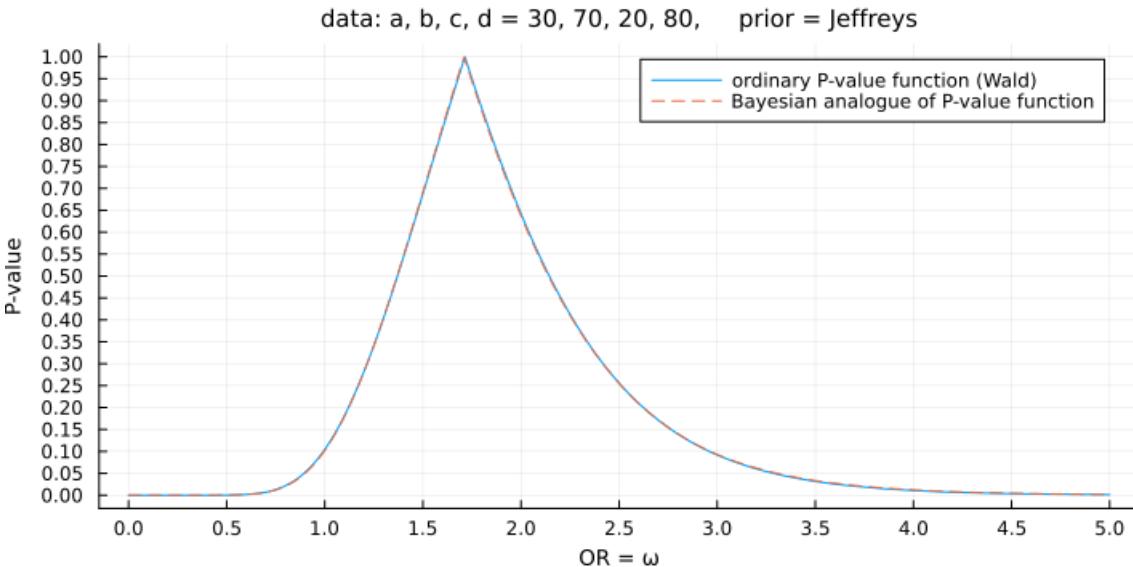
In [35]:

```

1 a, b, c, d = 30, 70, 20, 80
2 plot(w → pvalue_or_wald(a, b, c, d; w);
3       label="ordinary P-value function (Wald)")
4 plot!(w → pvalue_or_bayesian(w);
5       label="Bayesian analogue of P-value function", ls=:dash)
6 plot!(xguide="OR = ω", yguide="P-value")
7 plot!(xtick=0:0.5:10, ytick=0:0.05:1)
8 title!("data: a, b, c, d = $a, $b, $c, $d, prior = Jeffreys")
9 plot!(size=(720, 350))

```

Out[35]:



サンプルサイズがもっと小さい場合にはここまでぴったり一致しないが、この場合には非常によく一致している。

この一致を見てしまった人が、「ベイズ統計とP値を使う統計学は水と油である」のような考え方をすることは難しいように思われる。

以上によって、通常のP値を使った統計分析をBayes統計の方法でどのようにすれば置き換えることができるかがわかった。

Bayes統計の方法では事前分布(prior)という新しい道具を使えるので、別の新しいことも可能になる。

次の論文では以上と本質的に同じ方法をオッズ比ではなくリスク比の場合に適用して、Bayes統計の方法を使って統計分析を行っている：

- <https://www.nejm.org/doi/full/10.1056/NEJMoa2115869> (<https://www.nejm.org/doi/full/10.1056/NEJMoa2115869>)

この論文ではBayes統計の方法を使っているが、通常のP値を使う方法でも実質的に同じ結果が得られる。

3 Welchの t 検定について

2つの群の平均の差に関するWelchの t 検定に付随するP値と信頼区間については

- 「検定と信頼区間: 平均の比較」に関するノート
(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/12%20Hypothesis%20testing%20and%20confidence%20%20Two%20means.ipynb>)

で非常に詳しく解説したが、手続きが複雑なので以下で再度まとめておくことにする。



3.1 Welchの t 検定のP値と信頼区間の定義

データは別の母集団Aから抽出されたサイズ m の標本 x_1, \dots, x_m と別の母集団Bから抽出されたサイズ n の標本 y_1, \dots, y_n であると仮定する。

母集団Aからのサイズ m の標本についてその標本平均が従う分布は中心極限定理によって正規分布で十分に近似されていると仮定する。母集団Bとサイズ n についても同様であると仮定しておく。

母集団Aの(真の)平均を μ_x と書き、母集団Bの(真の)平均を μ_y と書く。

標本 x_1, \dots, x_m の標本平均を \bar{x} と書き、標本 y_1, \dots, y_n の標本平均を \bar{y} と書く。

母集団Aから一意的に決まっている定数 μ_x と標本を取り直すごとに確率的に変化する標本平均 \bar{x} を混同しないように注意せよ。 μ_y と \bar{y} についても同様に混同しないようにして欲しい。

さらに、標本 x_1, \dots, x_m の不偏分散を s_x^2 と書き、標本 y_1, \dots, y_n の不偏分散を s_y^2 と書く：

$$s_x^2 = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})^2, \quad s_y^2 = \frac{1}{n-1} \sum_{j=1}^n (y_j - \bar{y})^2.$$

これらを、母集団Aの(真の)分散 σ_x^2 と母集団Bの(真の)分散 σ_y^2 と混同しないように注意せよ。

以上の状況で、任意に与えられた実数 $\Delta\mu$ に対して、 $\mu_x - \mu_y = \Delta\mu$ という仮説のP値を定義しよう。

まず、t 統計量 $t = t(\Delta\mu)$ を次のように定義する：

$$t = t(\Delta\mu) = \frac{(\bar{x} - \bar{y}) - \Delta\mu}{\sqrt{\frac{s_x^2}{m} + \frac{s_y^2}{n}}}.$$

さらに、自由度と呼ばれる統計量 v を次のように定義する：

$$v = \frac{\left(\frac{s_x^2}{m} + \frac{s_y^2}{n}\right)^2}{\frac{(s_x^2/m)^2}{m-1} + \frac{(s_y^2/n)^2}{n-1}}.$$

そして、P値を次のように定義する：

$$\text{pvalue}_{\text{Welch}}(\bar{x}, \bar{y}, s_x^2, s_y^2 | m, n, \mu_x - \mu_y = \Delta\mu) = 2(1 - \text{cdf}(\text{TDist}(v), |t(\Delta\mu)|)).$$

ここで、 $\text{cdf}(\text{TDist}(v), x)$ は自由度 v の t 分布の累積分布函数(cumulative distribution function, cdf)である。

以上に対応する平均の差 $\mu_x - \mu_y$ に関する信頼度 $1 - \alpha$ の信頼区間は次のようになる：

$$\begin{aligned} & \text{confint}_{\text{Welch}}(\bar{x}, \bar{y}, s_x^2, s_y^2 | m, n, \alpha) \\ &= \left[\bar{x} - \bar{y} - t_{v, \alpha/2} \sqrt{\frac{s_x^2}{m} + \frac{s_y^2}{n}}, \bar{x} - \bar{y} + t_{v, \alpha/2} \sqrt{\frac{s_x^2}{m} + \frac{s_y^2}{n}} \right]. \end{aligned}$$

注意: 自由度 v をそのように定義する理由については

- 「検定と信頼区間: 平均の比較」のノート
(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/12%20Hypothesis%20testing%20and%20confidence%20%20Two%20means.ipynb>)

の「Welchのt検定で使う自由度の式の導出」の節を参照せよ。さらにその「Studentのt検定とWelchのt検定で使用するt分布の自由度の比較」の節では

$$(m+n-2) - v = \frac{\left(\frac{s_x^2}{m(m-1)} - \frac{s_y^2}{n(n-1)}\right)^2}{\frac{(s_x^2/(m(m-1)))^2}{n-1} + \frac{(s_y^2/(n(n-1)))^2}{m-1}} \geq 0$$

となることも注意している。これより $v \leq m+n-2$ となっている。

```
In [36]: 1 @vars s²_x s²_y m n v
2 v = (s²_x/m + s²_y/n)² / ((s²_x/m)²/(m-1) + (s²_y/n)²/(n-1))
3 lhs = (m+n-2) - v
4 rhsnum = (s²_x/(m*(m-1)) - s²_y/(n*(n-1)))²
5 rhstden = (s²_x/(m*(m-1)))²/(n-1) + (s²_y/(n*(n-1)))²/(m-1)
6 @show (lhs - rhsnum/rhstden).simplify()
7 Eq(lhs, rhsnum/rhstden)
```

```
(lhs - rhsnum / rhstden).simplify() = 0
```

```
Out[36]:  $m+n-\frac{\left(\frac{s^2_y}{n}+\frac{s^2_x}{m}\right)^2}{\frac{s^2_y^2}{n^2(n-1)}+\frac{s^2_x^2}{m^2(m-1)}}-2=\frac{\left(-\frac{s^2_y}{n(n-1)}+\frac{s^2_x}{m(m-1)}\right)^2}{\frac{s^2_y^2}{n^2(m-1)(n-1)}+\frac{s^2_x^2}{m^2(m-1)^2(n-1)}}$ 
```

注意: X_1, \dots, X_n は分散 σ^2 を持つ分布の標本で、 \bar{X} はその標本平均であるとする。このとき、不偏分散を計算するときに n ではなく、 $n-1$ で割る理由は単に

$$E \left[\sum_{i=1}^n (X_i - \bar{X})^2 \right] = (n-1)\sigma^2$$

となるからである。両辺を n ではなく、 $n-1$ で割らないと σ^2 が期待値として得られない。このような事実に関するより深い理解は 線形回帰が直交射影そのものであること を学べば得られる。線形回帰に一般化された場合の不偏分散は、 n 次元空間の中の r 次元部分空間の直交補空間の次元 $n-r$ で割ることによって得られる。詳しい説明は

- 「回帰(regression)」に関するノート
(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/14%20Regression.ipynb>)

の「 β と σ^2 の不偏推定量」の節にある。通常の標本の不偏分散は部分空間が1次元の $\{(t, t, \dots, t) \mid t \in \mathbb{R}\}$ の場合にになっている。

3.2 Welchのt検定のP値や信頼区間の計算例

以下の例は

- 「検定と信頼区間: 平均の比較」に関するノート
(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/12%20Hypothesis%20testing%20and%20confidence%20%20Two%20means.ipynb#Welch%E3%81%AE-t-%E6%A4%9C%E5%AE%9A%E3%81%AEP%E5%80%A4%E3%81%A8%E4%BF%A1%E9%A0%BC%E5%8C%BA%E9%>)

における

- 必修問題: Welchのt検定のP値と信頼区間の計算
(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/12%20Hypothesis%20testing%20and%20confidence%20%20Two%20means.ipynb#E5%BF%85%E4%BF%AE%E5%95%8F%E9%A1%8C:-Welch%E3%81%AE-t-%E6%A4%9C%E5%AE%9A%E3%81%AEP%E5%80%A4%E3%81%A8%E4%BF%A1%E9%A0%BC%E5%8C%BA%E9%>)

の再掲である。

サイズが $m = 20$, $n = 30$ のデータ

```

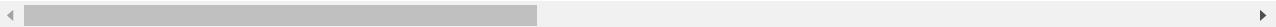
x = [19.2, 22.7, 7.8, 138.5, 70.5, 44.3, 84.0, 35.6, 72.4, 23.9,
      11.7, 26.6, 73.8, 118.3, 54.2, 57.6, 40.5, 117.4, 102.3, 67.6]

y = [44.3, 66.9, 62.9, 78.4, 71.2, 32.5, 111.4, 38.2, 68.2, 50.7,
      74.5, 46.2, 65.7, 58.7, 42.5, 57.4, 63.0, 67.9, 72.1, 117.7,
      124.1, 48.9, 91.8, 80.8, 60.2, 76.8, 76.3, 59.9, 70.7, 46.4]

```

について以下を求めよ.

(1) 仮説 $\mu_x - \mu_y = 0$ のP値.



3.2.1 WolframAlphaによるWelchのt検定のP値と信頼区間の計算の必修問題の解答例

WolframAlphaで標本の標本平均は `mean {a, b, c, ...}` のように入力すれば計算できる.

WolframAlphaで標本の不偏分散は `var {a, b, c, ...}` のように入力すれば計算できる.

注意: 統計学の慣習では、標本の分散は n ではなく $n - 1$ で割る方の不偏分散を意味することが多く、ソフトウェア側でもその慣習に従っていることが多い。WolframAlphaやJuliaやRの `var` はデフォルトで不偏分散を計算するが、Pythonのnumpyの `np.var` はそうではないので注意せよ。`np.var(x, ddof=1)` のように `ddof=1` が必要になる。詳しくは公式ドキュメントの

- [\(https://numpy.org/doc/stable/reference/generated/numpy.var.html\)](https://numpy.org/doc/stable/reference/generated/numpy.var.html)

を参照せよ。コンピュータで標本の分散の値を計算する場合にはそのための函数の仕様を確認しておいた方がよい。**注意終**

$\bar{x} \rightarrow \text{mean } \{19.2, 22.7, 7.8, 138.5, 70.5, 44.3, 84.0, 35.6, 72.4, 23.9, 11.7, 26.6, 73.8, 118.3, 54.2, 57.6, 40.5, 117.4, 102.3, 67.6\} \rightarrow \text{実行 (https://www.wolframalpha.com/input?i=mean%7B19.2%2C+22.7%2C+7.8%2C+138.5%2C+70.5%2C+44.3%2C+84.0%2C+35.6%2C+72.4%2C+23.9%2C+11.7%2C+59.445)}$

$s_x^2 \rightarrow \text{var } \{19.2, 22.7, 7.8, 138.5, 70.5, 44.3, 84.0, 35.6, 72.4, 23.9, 11.7, 26.6, 73.8, 118.3, 54.2, 57.6, 40.5, 117.4, 102.3, 67.6\} \rightarrow \text{実行 (https://www.wolframalpha.com/input?i=var%7B19.2%2C+22.7%2C+7.8%2C+138.5%2C+70.5%2C+44.3%2C+84.0%2C+35.6%2C+72.4%2C+23.9%2C+11.7%2C+479.481)}$

$\bar{y} \rightarrow \text{mean } \{44.3, 66.9, 62.9, 78.4, 71.2, 32.5, 111.4, 38.2, 68.2, 50.7, 74.5, 46.2, 65.7, 58.7, 42.5, 57.4, 63.0, 67.9, 72.1, 117.7, 124.1, 48.9, 91.8, 80.8, 60.2, 76.8, 76.3, 59.9, 70.7, 46.4\} \rightarrow \text{実行 (https://www.wolframalpha.com/input?i=mean%7B44.3%2C+66.9%2C+62.9%2C+78.4%2C+71.2%2C+32.5%2C+111.4%2C+38.2%2C+68.2%2C+50.7%2C+74.5%2C+67.5433)}$

$s_y^2 \rightarrow \text{var } \{44.3, 66.9, 62.9, 78.4, 71.2, 32.5, 111.4, 38.2, 68.2, 50.7, 74.5, 46.2, 65.7, 58.7, 42.5, 57.4, 63.0, 67.9, 72.1, 117.7, 124.1, 48.9, 91.8, 80.8, 60.2, 76.8, 76.3, 59.9, 70.7, 46.4\} \rightarrow \text{実行 (https://www.wolframalpha.com/input?i=var%7B44.3%2C+66.9%2C+62.9%2C+78.4%2C+71.2%2C+32.5%2C+111.4%2C+38.2%2C+68.2%2C+50.7%2C+74.5%2C+479.481)}$

以下において、 x, v, y, w はそれぞれ $\bar{x}, s_x^2, \bar{y}, s_y^2$ を意味する。

自由度 $v \rightarrow (v/m+w/n)^2 / ((v/m)^2/(m-1)+(w/n)^2/(n-1)) \text{ where } \{m=20, x=59.445, v=1448.48, n=30, y=67.5433, w=479.481\} \rightarrow \text{実行 (https://www.wolframalpha.com/input?i=%28v%2Fm%2Bw%2Fn%29%5E2%2F%28v%2Fm%29%5E2%2F%28m-1%29%2B%28w%2Fm%29%5E2%2F%28n-1%29%29+where+%7Bm%3D20%2C+x%3D59.445%2C+v%3D1448.48%2C+n%3D30%2C+y%3D67.5433%2C+w%3D479.481)}$

(1)

仮説 $\mu_x - \mu_y = 0$ の t 値 $\rightarrow (x-y-0) / \sqrt{v/m+w/n}$ where $\{m=20, x=59.445, v=1448.48, n=30, y=67.5433, w=479.481\} \rightarrow \text{実行 (https://www.wolframalpha.com/input?i=28x-28y-%29%2Fsqrt%28v%2Fm%2Bw%2Fn%29+where+%7Bm%3D20%2C+x%3D59.445%2C+v%3D1448.48%2C+n%3D30%2C+y%3D67.5433%2C+w%3D479.481)}$

仮説 $\mu_x - \mu_y = 0$ のP値 $\rightarrow 2(1 - \text{CDF}(TDistribution}(27.4358), 0.861294)) \rightarrow \text{実行 (https://www.wolframalpha.com/input?i=2%281+-+CDF%28TDistribution%2827.4358%29%2C+0.861294%29%29)}$

(2)

仮説 $\mu_x - \mu_y = -30$ の t 値 $\rightarrow (x-y+30)/\sqrt{v/m+w/n}$ where $\{m=20, x=59.445, v=1448.48, n=30, y=67.5433, w=479.481\} \rightarrow$ 実行 ([https://www.wolframalpha.com/input/?i=\(x-y+30\)/sqrt\(v/m+w/n\)+where+m=20+x=59.445+v=1448.48+n=30+y=67.5433+w=479.481](https://www.wolframalpha.com/input/?i=(x-y+30)/sqrt(v/m+w/n)+where+m=20+x=59.445+v=1448.48+n=30+y=67.5433+w=479.481)) $\rightarrow 2.32935$

仮説 $\mu_x - \mu_y = -30$ のP値 $\rightarrow 2(1 - cdf(TDistribution(27.4358), 2.32935)) \rightarrow$ 実行 ([https://www.wolframalpha.com/input/?i=2*\(1-cdf\(TDistribution\(27.4358\), 2.32935\)\)](https://www.wolframalpha.com/input/?i=2*(1-cdf(TDistribution(27.4358), 2.32935)))) $\rightarrow 0.0274389$

(3)

$\alpha = 0.05$ のときの $t_{v,\alpha/2} \rightarrow quantile(TDistribution(27.4358), 0.975) \rightarrow$ 実行 ([https://www.wolframalpha.com/input/?i=quantile\(TDistribution\(27.4358\), 0.975\)](https://www.wolframalpha.com/input/?i=quantile(TDistribution(27.4358), 0.975))) $\rightarrow 2.05031$

$\mu_x - \mu_y$ の 95% 信頼区間 $\rightarrow \{x-y-2.05031*\sqrt{v/m+w/n}, x-y+2.05031*\sqrt{v/m+w/n}\}$ where $\{m=20, v=1448.48, w=479.481\} \rightarrow$ 実行 ([https://www.wolframalpha.com/input/?i={x-y-2.05031*sqrt\(v/m+w/n\), x-y+2.05031*sqrt\(v/m+w/n\)}+where+m=20+v=1448.48+w=479.481](https://www.wolframalpha.com/input/?i={x-y-2.05031*sqrt(v/m+w/n), x-y+2.05031*sqrt(v/m+w/n)}+where+m=20+v=1448.48+w=479.481))

3.2.2 Julia言語によるWelchのt検定のP値と信頼区間の計算の必修問題の解答例

まず、定義通りにJulia言語のコードで必要な函数を書く。

Julia言語対応環境において

- \bar{x} は `x\bar` と入力してタブキーを押せば入力できる。
- sx^2 は `sx\^2` と入力してタブキーを押せば入力できる。
- Δ は `\Delta` と入力してタブキーを押せば入力できる。
- v は `\nu` と入力してタブキーを押せば入力できる。
- \sqrt は `\sqrt` と入力してタブキーを押せば入力できる。

ユニコードの文字をプログラムのコードとして使いたくない人は \bar{x} や sx^2 の代わりに `xbar`, `sx2` のように書いても全然問題ない。

以下ではほとんど数式通りにコードが書かれている

In [37]:

```

1 # 確率分布を扱う場合には常に以下を前もって実行しておく。
2
3 using Distributions
4
5 # t値の計算の仕方 (デフォルトで Δμ = 0 にしておく)
6
7 function tvalue_welch(m, x̄, sx², n, ȳ, sy²; Δμ=0)
8     (x̄ - ȳ - Δμ) / √(sx²/m + sy²/n)
9 end
10
11 function tvalue_welch(x, y; Δμ=0)
12     m, x̄, sx² = length(x), mean(x), var(x)
13     n, ȳ, sy² = length(y), mean(y), var(y)
14     tvalue_welch(m, x̄, sx², n, ȳ, sy²; Δμ)
15 end
16
17 # 自由度 ν の計算の仕方
18
19 function degree_of_freedom_welch(m, sx², n, sy²)
20     (sx²/m + sy²/n)^2 / ((sx²/m)^2/(m-1) + (sy²/n)^2/(n-1))
21 end
22
23 function degree_of_freedom_welch(x, y)
24     m, sx² = length(x), var(x)
25     n, sy² = length(y), var(y)
26     degree_of_freedom_welch(m, sx², n, sy²)
27 end
28
29 # P値の計算の仕方
30
31 function pvalue_welch(m, x̄, sx², n, ȳ, sy²; Δμ=0)
32     t = tvalue_welch(m, x̄, sx², n, ȳ, sy²; Δμ)
33     ν = degree_of_freedom_welch(m, sx², n, sy²)
34     2ccdf(TDist(ν), abs(t))
35 end
36
37 function pvalue_welch(x, y; Δμ=0)
38     m, x̄, sx² = length(x), mean(x), var(x)
39     n, ȳ, sy² = length(y), mean(y), var(y)
40     pvalue_welch(m, x̄, sx², n, ȳ, sy²; Δμ)
41 end
42
43 # 信頼区間の計算の仕方 (デフォルトでは α = 0.05 にしておく)
44
45 function confint_welch(m, x̄, sx², n, ȳ, sy²; α=0.05)
46     ν = degree_of_freedom_welch(m, sx², n, sy²)
47     c = quantile(TDist(ν), 1-α/2)
48     SEhat = √(sx²/m + sy²/n)
49     [x̄-ȳ-c*SEhat, x̄-ȳ+c*SEhat]
50 end
51
52 function confint_welch(x, y; α=0.05)
53     m, x̄, sx² = length(x), mean(x), var(x)
54     n, ȳ, sy² = length(y), mean(y), var(y)
55     confint_welch(m, x̄, sx², n, ȳ, sy²; α)
56 end

```

Out[37]: confint_welch (generic function with 2 methods)

以上のコードを何も考えずにコピー&ペーストして使っても勉強にならない。

可能ならば全部自分で書き直してみるべきである。

そのためには、出て来たビルトイン函数のすべてについて公式ドキュメントを参照した方がよい。

Juliaの場合には公式ドキュメントはほぼ ? で表示されるドキュメントに等しい。

Julia対応環境では例えば ?TDist と入力すれば TDist のドキュメントを読むことができる。

In [38]: 1 ?TDist

```
search: TDist tdistpdf tdistcdf tdistccdf tdistlogpdf tdistlogcdf tdistinvcdf
```

Out[38]: TDist(v)

The Students *T distribution* with v degrees of freedom has probability density function

$$f(x; v) = \frac{1}{\sqrt{v}B(1/2, v/2)} \left(1 + \frac{x^2}{v}\right)^{-\frac{v+1}{2}}$$

TDist(d) # *t-distribution with v degrees of freedom*

```
params(d) # Get the parameters, i.e. (v,)  
dof(d) # Get the degrees of freedom, i.e. v
```

External links

[Student's T distribution on Wikipedia \(https://en.wikipedia.org/wiki/Student%27s_t-distribution\)](https://en.wikipedia.org/wiki/Student%27s_t-distribution)

In [39]: 1 # x, y に標本の値を代入する.

```
2 x = [19.2, 22.7, 7.8, 138.5, 70.5, 44.3, 84.0, 35.6, 72.4, 23.9,  
3 11.7, 26.6, 73.8, 118.3, 54.2, 57.6, 40.5, 117.4, 102.3, 67.6];  
4 y = [44.3, 66.9, 62.9, 78.4, 71.2, 32.5, 111.4, 38.2, 68.2, 50.7,  
5 74.5, 46.2, 65.7, 58.7, 42.5, 57.4, 63.0, 67.9, 72.1, 117.7,  
6 124.1, 48.9, 91.8, 80.8, 60.2, 76.8, 76.3, 59.9, 70.7, 46.4];
```

In [40]: 1 # (1)

```
2 pval1 = pvalue_welch(x, y; Δμ = 0)
```

Out[40]: 0.39653998689489345

統計学関係のコードを自分で書いた場合には、何らかの方法で実装を間違っていないことを確認した方がよい。そのための1つの方法は別の方法で計算して確認することである。例えばR言語で計算し直すと次のようになる。

In [41]: 1 @rput x y

```
2 pval1_R = rcopy(R"""t.test(x, y, mu = 0)""")[:p_value]
```

Out[41]: 0.3965399868948938

浮動小数点計算で生じる微小な誤差を除いてよく一致している。

In [42]: 1 pval1 - pval1_R

Out[42]: -3.3306690738754696e-16

注意: Julia言語から[R言語 \(https://cran.r-project.org/\)](https://cran.r-project.org/)を使うためには、自分のパソコンにR言語一式をインストールし、さらにJulia言語の側で

- [https://github.com/JuliaInterop/RCall.jl \(https://github.com/JuliaInterop/RCall.jl\)](https://github.com/JuliaInterop/RCall.jl).

をインストールしておく必要がある。

統計学を勉強するためには、自分のパソコンでJulia, Python, Rの3つを使えるようにしておくと非常に便利である。しかし、そのためにはまた別のスキルが必要になる。数年計画で身に付けるようにしておくとよいと思われる。トラブルに出会ったら、表示されたエラーメッセージをインターネットで検索してみるとよい。コンピュータやプログラミングでは英語で書かれたトラブルの解決法を読むことは普通であり、そのために英語にも慣れておくことが必要になる。結局のところ、統計学を十分にマスターするために、コンピュータの使い方と技術英語もマスターする必要が生じる。しかし、統計学を含めたそれら諸々のスキルを身に付けた人達は我々の社会の中で少数派であり、学生時代のうちに身に付けることに成功した人達は相當に自信を持つてよいと思う。繰り返しになるが、数年計画で無理せずに身に付ければ十分である。あせる必要はない。就職した後にも勉強を続けることが普通である。注意終

WolframAlphaで求めた値の相対誤差が十分に小さいことの確認。

In [43]: 1 0.396541/pval1 - 1

Out[43]: 2.5548624098536976e-6

```
In [44]: 1 # (2)
2 pval2 = pvalue_welch(x, y; Δμ = -30)
```

```
Out[44]: 0.027439073239531347
```

R言語で計算し直すと次のようになる。

```
In [45]: 1 @rput x y
2 pval2_R = rcopy(R"""\t.test(x, y, mu = -30)""")[:p_value]
```

```
Out[45]: 0.027439073239531344
```

```
In [46]: 1 pval2 - pval2_R
```

```
Out[46]: 3.469446951953614e-18
```

WolframAlphaで求めた値の相対誤差が十分に小さいことの確認。

```
In [47]: 1 0.0274389/pval2 - 1
```

```
Out[47]: -6.313607235797214e-6
```

```
In [48]: 1 # (3)
2 ci3 = confint_welch(x, y; α = 0.05)
3 print(ci3)
```

```
[-27.37632153461527, 11.179654867948582]
```

R言語で計算し直すと次のようになる。

```
In [49]: 1 @rput x y
2 ci3_R = rcopy(R"""\t.test(x, y, conf.level = 0.95)""")[:conf_int]
3 print(ci3_R)
```

```
[-27.376321534615265, 11.179654867948592]
```

```
In [50]: 1 ci3 - ci3_R ▶ print
```

```
[-3.552713678800501e-15, -1.0658141036401503e-14]
```

WolframAlphaで求めた値の相対誤差が十分に小さいことの確認。

```
In [51]: 1 [-27.3763, 11.1797] ./ ci3 .- 1 ▶ print
```

```
[-7.866146385371309e-7, 4.03698074324943e-6]
```

Julia言語では要素ごとの演算(broadcast)にするためにはこのように演算記号に . を付ける。

せっかくなのでP値函数もプロットしておこう。

In [52]:

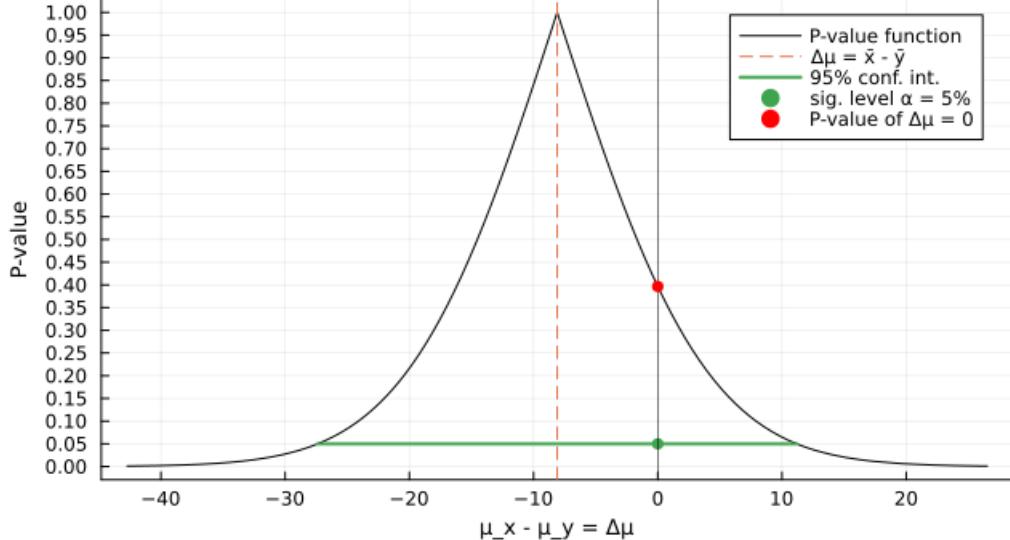
```

1 @show xlim = confint_welch(x, y; α=1e-3)
2 plot(Δμ → pvalue_welch(x, y; Δμ), xlim...; label="P-value function", c=:black)
3 vline!([mean(x) - mean(y)]; label="Δμ = x̄ - ȳ", ls=:dash, c=2)
4 plot!(ci3, fill(0.05, 2); label="95% conf. int.", lw=2, c=3)
5 scatter!([0], [0.05]; label="sig. level α = 5%", c=3, msc=:auto)
6 vline!([0]; label="", lw=0.5, c=:black)
7 scatter!([0], [pval1]; label="P-value of Δμ = 0", c=:red, msc=:auto)
8 plot!(xguide="μ_x - μ_y = Δμ", yguide="P-value")
9 plot!(ytick=0:0.05:1)
10 plot!(size=(640, 350))

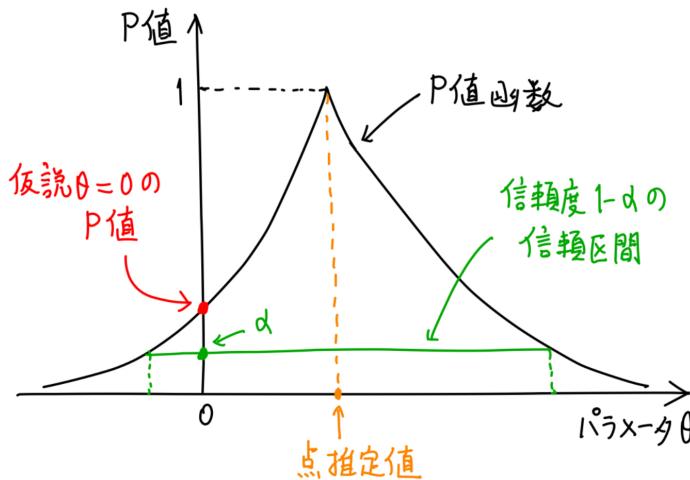
```

xlim = confint_welch(x, y; α = 0.001) = [-42.72395560198741, 26.527288935320726]

Out[52]:



これと以下を比較せよ.



上のWelchの t 検定でのP値函数のグラフでは、パラメータ θ は2つの群の平均の差を意味する $\Delta\mu$ になっている。

そして、点推定値は $\Delta\mu = \bar{x} - \bar{y}$ になっている。(この点推定値は正規分布の標本分布モデルにおける最尤推定値にもなっている。)

母集団からの標本の無作為抽出(random sampling, ランダム・サンプリング)を行うごとに変化するデータの数値の確率的揺らぎも考慮すると、標本を取得した2つの母集団の平均の差 $\mu_x - \mu_y$ (これの値は未知)がぴったり点推定値 $\bar{x} - \bar{y}$ になっているとは考えられないことに注意せよ。

データの数値と $\mu_x - \mu_y = \Delta\mu$ という仮説の整合性がP値函数のグラフを見ればわかる。

大雑把に、 $\Delta\mu$ の値が -43 未満だったり、27 より大きいとき、 $\mu_x - \mu_y = \Delta\mu$ という仮説とデータの整合性はほぼないように見える。(この判断は有意水準を $\alpha = 0.1\%$ とした場合に相当している。)

さらに大雑把に閾値 $\alpha = 5\%$ で切断した場合の信頼区間は大体 $[-27, 11]$ でそれよりは狭くなっている。

4 数学的な補足: 大数の法則と中心極限定理について

4.1 二項分布の大数の法則

二項分布

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

に関する大数の法則は、 n が大きなとき、 k/n の分布が p の近くに集中することを意味する。これは

$$\lim_{n \rightarrow \infty} \sum_{k=0}^n f\left(\frac{k}{n}\right) \binom{n}{k} p^k (1-p)^{n-k} = f(p)$$

が成立することを意味している。

注意: この結果の厳密な直接的証明は

- 高木貞治『解析概論』岩波書店

の「§78. Weierstrassの定理」の節にある。その式(8)を見よ。すなわち、閉区間上の連続函数が多項式函数で一様近似できるというWeierstrassの定理は二項分布の大数の法則(の精密化)から得られる。この方法は Serge Bernstein による。注意終

$f(x) = \cos(\pi x)$, $p = 1/3$ の場合について、二項分布の大数の法則を数値的に確認してみよう。

In [53]:

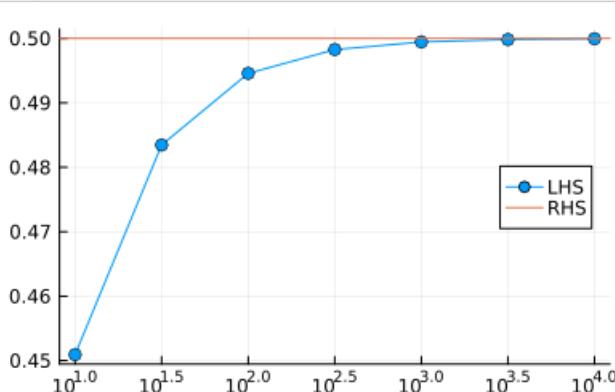
```
1 f(x) = cospi(x)
2 n, p = 10^6, 1/3
3 @show bin = Binomial(n, p)
4 @show LHS = sum(f(k/n)*pdf(bin, k) for k in support(bin))
5 @show RHS = f(p)
6 @show LHS/RHS - 1;

bin = Binomial(n, p) = Binomial{Float64}(n=1000000, p=0.3333333333333333)
LHS = sum((f(k / n) * pdf(bin, k) for k = support(bin))) = 0.49999945168896076
RHS = f(p) = 0.5
LHS / RHS - 1 = -1.096622078478049e-6
```

In [54]:

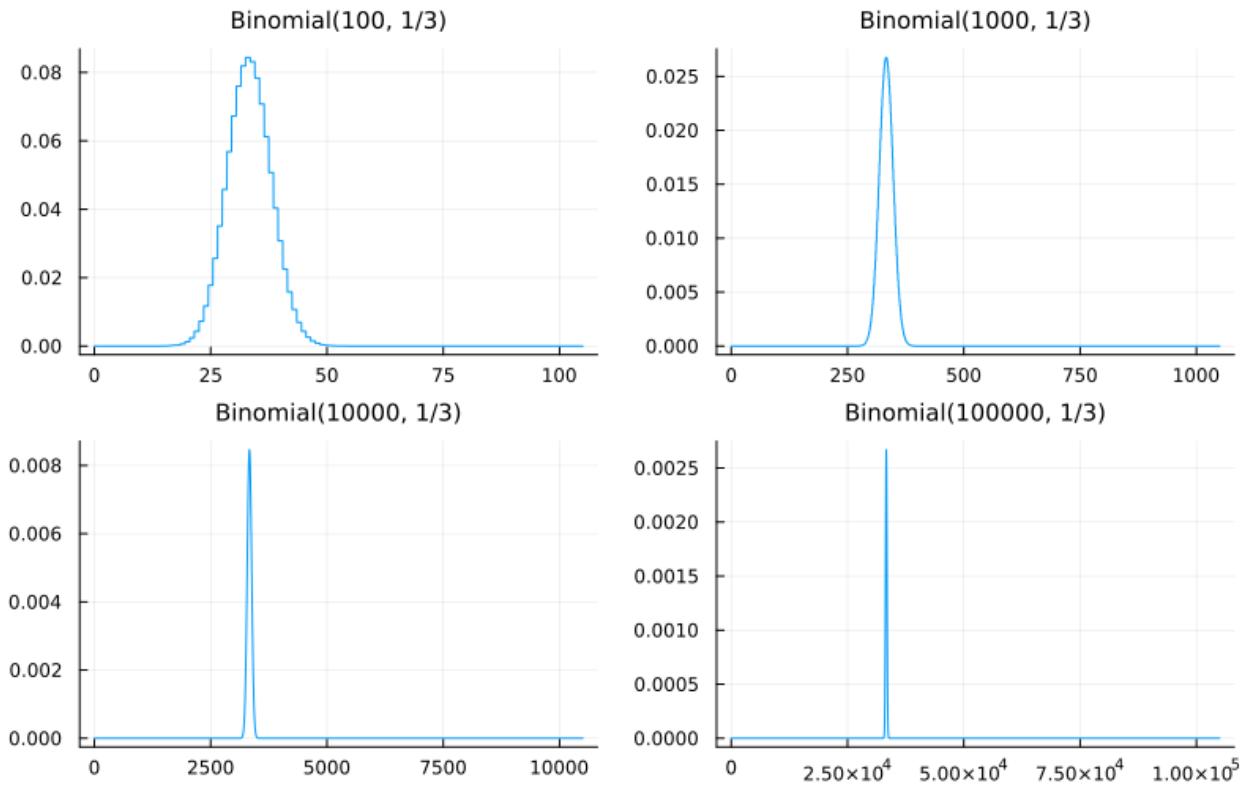
```
1 f(x) = cospi(x)
2 ns, p = 10 .^ (1:0.5:4), 1/3
3 LHSs = [(n = round(Int, n); bin = Binomial(n, p);
4           sum(f(k/n)*pdf(bin, k) for k in support(bin))) for n in ns]
5 RHS = f(p)
6 plot(ns, LHSs; xscale=:log10, xtick=ns, marker=:o,
7       label="LHS", legend=:right)
8 hline!([RHS]; label="RHS")
```

Out[54]:



```
In [55]: 1 plot(plot(x → pdf(Binomial(10^k, 1/3), round(Int, x)), 0, 1.05*10^k;
2           label="", title="Binomial($(10^k), 1/3)") for k in 2:5)...;
3           size=(800, 500), layout=(2,2))
```

Out[55]:



4.2 二項分布の中心極限定理

二項分布

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

に関する大数の法則は、 n が大きなとき、 k の分布が二項分布と同じ平均 np と分散 $np(1-p)$ を持つ正規分布で近似されることを意味する。すなわち、 n が大きなとき、

$$z = \frac{k - np}{\sqrt{np(1-p)}}$$

の分布が標準正規分布で近似されることを意味する。このことは次が成立することを意味する：

$$\lim_{n \rightarrow \infty} \sum_{k=0}^n f\left(\frac{k - np}{\sqrt{np(1-p)}}\right) \binom{n}{k} p^k (1-p)^{n-k} = \int_{-\infty}^{\infty} f(z) \frac{e^{-z^2/2}}{\sqrt{2\pi}} dz.$$

$f(x) = \cos(x)$, $p = 1/3$ の場合について、二項分布の中心極限定理を数値的に確認してみよう。

```
In [56]: 1 f(x) = cos(x)
2 n, p = 10^6, 1/3
3 @show bin = Binomial(n, p)
4 @show LHS = sum(f((k-n*p)/sqrt(n*p*(1-p)))*pdf(bin, k) for k in support(bin))
5 g(z) = f(z) * pdf(Normal(0, 1), z)
6 @show RHS = quadgk(g, -Inf, Inf)[1] # using QuadGK すると使える数値積分函数
7 @show LHS/RHS - 1;
```

```
bin = Binomial(n, p) = Binomial{Float64}(n=1000000, p=0.3333333333333333)
LHS = sum((f((k - n * p) / sqrt(n * p * (1 - p))) * pdf(bin, k) for k = support(bin))) = 0.60653
06175921816
RHS = (quadgk(g, -Inf, Inf))[1] = 0.606530659712633
LHS / RHS - 1 = -6.944488406546157e-8
```

実は次が成立している：

$$\int_{-\infty}^{\infty} \cos(tz) \frac{e^{-z^2/2}}{\sqrt{2\pi}} dz = \int_{-\infty}^{\infty} e^{itz} \frac{e^{-z^2/2}}{\sqrt{2\pi}} dz = E[e^{itz}] = e^{-t^2/2}.$$

ここで Z は標準正規分布に従う確率変数である。

```
In [57]: 1 @vars z t real=true
2 g_sym(z) = cos(t*z) * exp(-z^2/2)/sqrt(2*pi)
3 expr = sympy.Integral(g_sym(z), (z, -oo, oo))
4 sol_exact = expr.doit()
5 Eq(expr, sol_exact)
```

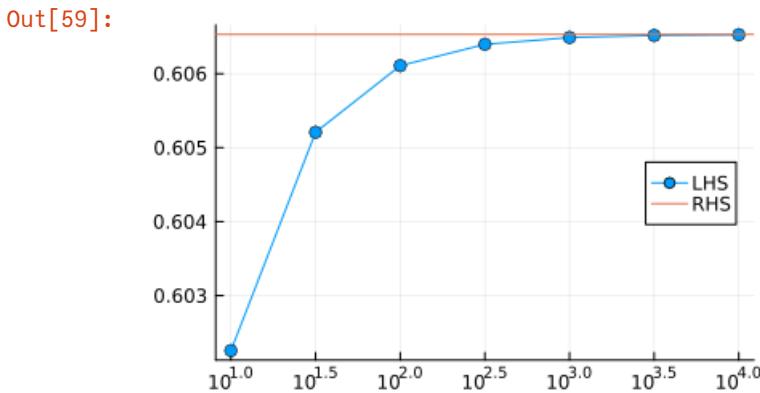
```
Out[57]: 
$$\int_{-\infty}^{\infty} \frac{\sqrt{2}e^{-\frac{z^2}{2}} \cos(tz)}{2\sqrt{\pi}} dz = e^{-t^2/2}$$

```

```
In [58]: 1 @show sol_numeriacal = float(sol_exact(t=1))
2 @show RHS/sol_numeriacal - 1;
```

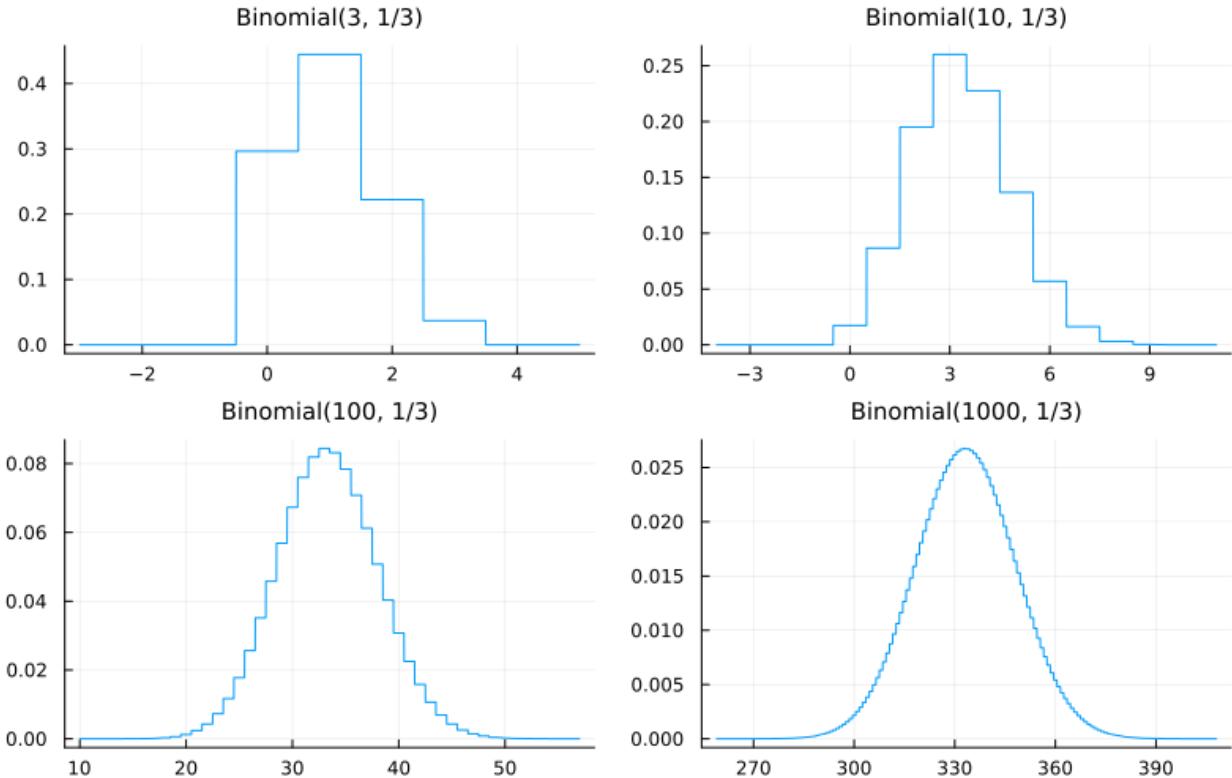
```
sol_numeriacal = float(sol_exact(t=1)) = 0.6065306597126334
RHS / sol_numeriacal - 1 = -7.771561172376096e-16
```

```
In [59]: 1 f(x) = cos(x)
2 ns, p = 10 .^(1:0.5:4), 1/3
3 LHSs = [(n = round(Int, n); bin = Binomial(n, p);
4           sum(f((k-n*p))/sqrt(n*p*(1-p)))*pdf(bin, k)) for k in support(bin)) for n in ns]
5 g(z) = f(z) * pdf(Normal(0, 1), z)
6 RHS = quadgk(g, -Inf, Inf)[1] # using QuadGK すると使える数値積分函数
7 plot(ns, LHSs; xscale=:log10, xtick=ns, marker=:o,
8       label="LHS", legend=:right)
9 hline!([RHS]; label="RHS")
```



```
In [60]: 1 plot(plot(x → pdf(Binomial(n, 1/3), round(Int, x)),
2           round(Int, n/3 - 5\sqrt(n*1/3*2/3)),
3           round(Int, n/3 + 5\sqrt(n*1/3*2/3));
4           label="", title="Binomial($n, 1/3)") for n in (3, 10, 100, 1000))...
5           size=(800, 500), layout=(2,2))
```

Out[60]:



4.3 他の分布の場合

正規分布で近似される他の分布の場合にも同様の結果が成立している。

4.3.1 例: Poisson分布

$$P(k|\lambda) = \frac{e^{-\lambda} \lambda^k}{k!} \quad (k = 0, 1, 2, \dots)$$

の平均と分散は共に λ であり, λ が大きなとき同じ平均と分散を持つ正規分布で近似されるので,

$$\lim_{\lambda \rightarrow \infty} \sum_{k=0}^{\infty} f\left(\frac{k-\lambda}{\sqrt{\lambda}}\right) \frac{e^{-\lambda} \lambda^k}{k!} = \int_{-\infty}^{\infty} f(z) \frac{e^{-z^2/2}}{\sqrt{2\pi}} dz.$$

```
In [61]: 1 f(x) = cos(x)
2 λ = 10^6
3 @show poi = Poisson(λ)
4 m, s = mean(poi), std(poi)
5 supp = max(0, round(Int, m-6s)):round(Int, m+6s)
6 @show LHS = sum(f((k-λ)/\sqrt{λ}) * pdf(poi, k) for k in supp)
7 g(z) = f(z) * pdf(Normal(0, 1), z)
8 @show RHS = quadgk(g, -Inf, Inf)[1] # using QuadGK すると使える数値積分函数
9 @show LHS/RHS - 1;
```

```
poi = Poisson(λ) = Poisson{Float64}(λ=1.0e6)
LHS = sum((f((k - λ) / √λ) * pdf(poi, k) for k = supp)) = 0.6065306740138549
RHS = (quadgk(g, -Inf, Inf))[1] = 0.606530659712633
LHS / RHS - 1 = 2.3578728747253308e-8
```

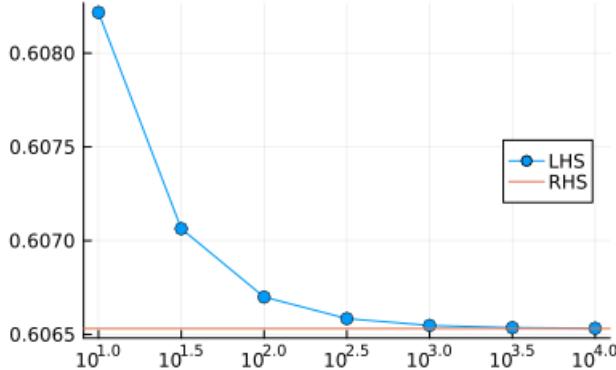
In [62]:

```

1 f(x) = cos(x)
2 λs = 10 .^ (1:0.5:4)
3 LHSs = [(poi = Poisson(λ));
4     (m, s) = (mean(poi), std(poi));
5     supp = max(0, round(Int, m-6s)):round(Int, m+6s);
6     sum(f((k-λ)/√λ) * pdf(poi, k) for k in supp)) for λ in λs]
7 g(z) = f(z) * pdf(Normal(0, 1), z)
8 RHS = quadgk(g, -Inf, Inf)[1] # using QuadGK すると使える数値積分函数
9 plot(λs, LHSs; xscale=:log10, xtick=λs, marker=:o,
10     label="LHS", legend=:right)
11 hline!([RHS]; label="RHS")

```

Out[62]:



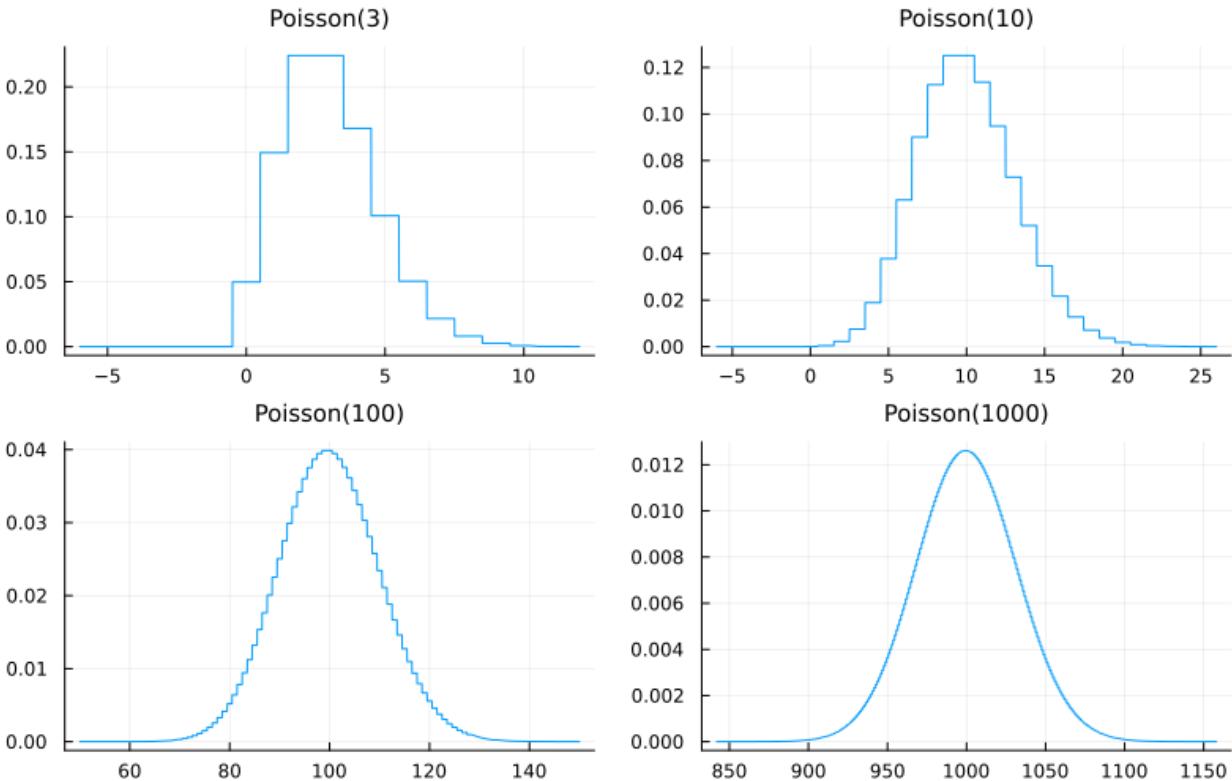
In [63]:

```

1 plot(plot(x → pdf(Poisson(λ), round(Int, x)),
2         round(Int, λ - 5√(λ)),
3         round(Int, λ + 5√(λ));
4         label="", title="Poisson($λ)") for λ in (3, 10, 100, 1000))...;
5 size=(800, 500), layout=(2,2))

```

Out[63]:



4.3.2 例: Gamma分布

$$p(x|\alpha, \theta) = \frac{e^{-x/\theta} x^{\alpha-1}}{\theta^\alpha \Gamma(\alpha)}$$

は平均 $\alpha\theta$, 分散 $\alpha\theta^2$ を持ち, α が大きなとき同じ平均と分散を持つ正規分布で近似されるので,

$$\lim_{\lambda \rightarrow \infty} \int_0^\infty f\left(\frac{x - \alpha\lambda}{\sqrt{\alpha\theta^2}}\right) \frac{e^{-x/\theta} x^{\alpha-1}}{\theta^\alpha \Gamma(\alpha)} dx = \int_{-\infty}^\infty f(z) \frac{e^{-z^2/2}}{\sqrt{2\pi}} dz.$$

In [64]:

```

1 f(x) = cos(x)
2 α, θ = 10^6, 2
3 @show gam = Gamma(α, θ)
4 F(x) = f((x - α*θ)/sqrt(α*θ^2)) * pdf(gam, x)
5 m, s = mean(gam), std(gam)
6 a, b = m-5s, m+5s
7 @show LHS = quadgk(F, a, b)[1]
8 g(z) = f(z) * pdf(Normal(0, 1), z)
9 @show RHS = quadgk(g, -Inf, Inf)[1] # using QuadGK すると使える数値積分函数
10 @show LHS/RHS - 1;

```

```

gam = Gamma(α, θ) = Gamma{Float64}(α=1.0e6, θ=2.0)
LHS = (quadgk(F, a, b))[1] = 0.6065305209355965
RHS = (quadgk(g, -Inf, Inf))[1] = 0.606530659712633
LHS / RHS - 1 = -2.2880465200270095e-7

```

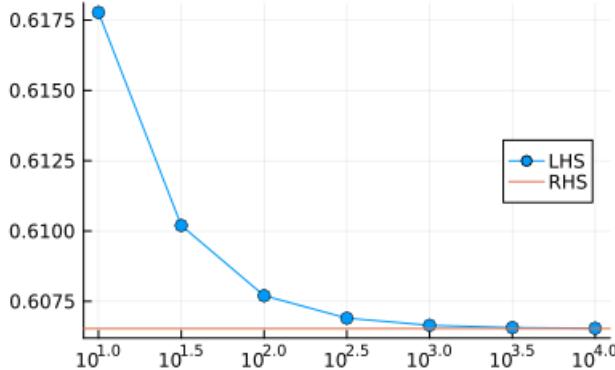
In [65]:

```

1 f(x) = cos(x)
2 αs, θ = 10 .^ (1:0.5:4), 2
3 LHSs = [(gam = Gamma(α, θ);
4     F(x) = f((x - α*θ)/sqrt(α*θ^2)) * pdf(gam, x));
5     (m, s) = (mean(gam), std(gam));
6     (a, b) = (m-5s, m+5s);
7     quadgk(F, a, b)[1]) for α in αs]
8 g(z) = f(z) * pdf(Normal(0, 1), z)
9 RHS = quadgk(g, -Inf, Inf)[1] # using QuadGK すると使える数値積分函数
10 plot(αs, LHSs; xscale=:log10, xtick=αs, marker=:o,
11       label="LHS", legend=:right)
12 hline!([RHS]; label="RHS")

```

Out[65]:



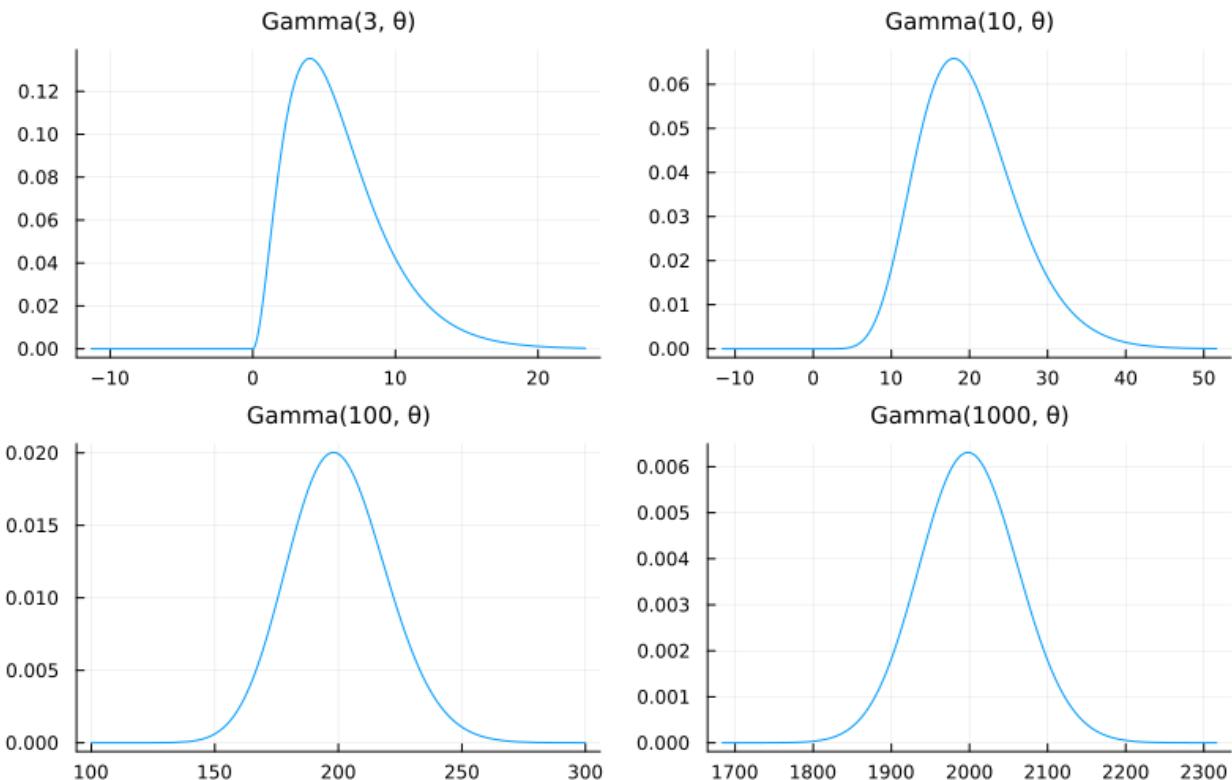
In [66]:

```

1 plot((plot(x → pdf(Gamma(α, θ), x), α*θ - 5sqrt(α*θ^2), α*θ + 5sqrt(α*θ^2);
2           label="", title="Gamma($α, θ)") for α in (3, 10, 100, 1000))...
3           size=(800, 500), layout=(2,2))

```

Out[66]:



4.4 標本分布の場合

平均 μ , 分散 σ^2 を持つ分布 D の標本分布 D^n に従う確率変数 (X_1, \dots, X_n) について,

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$$

は平均 μ , 分散 σ^2/n を持ち, 大数の法則より, n が大きなとき μ の近くにその分布は集中する. このことは次が成立することを意味している:

$$\lim_{n \rightarrow \infty} E[f(\bar{X}_n)] = f(\mu).$$

さらに, 中心極限定理より, \bar{X}_n は n が大きなとき同じ平均と分散を持つ正規分布に近似的に従う. すなわち,

$$Z_n = \frac{\bar{X}_n - \mu}{\sqrt{\sigma^2/n}} = \frac{\sqrt{n}(\bar{X}_n - \mu)}{\sigma} = \frac{1}{\sqrt{n}\sigma} \sum_{i=1}^n (X_i - \mu)$$

は n が大きなとき近似的に標準正規分布に従う. このことは次が成立することを意味している:

$$\lim_{n \rightarrow \infty} E[f(Z_n)] = \int_{-\infty}^{\infty} f(z) \frac{e^{-z^2/2}}{\sqrt{2\pi}} dz.$$

例えば, 平均 μ , 分散 σ^2 の分布 D が確率密度函数 $p(x)$ を持つとき,

$$\begin{aligned} \lim_{n \rightarrow \infty} \int \cdots \int f\left(\frac{x_1 + \cdots + x_n}{n}\right) p(x_1) \cdots p(x_n) dx_1 \cdots dx_n &= f(\mu), \\ \lim_{n \rightarrow \infty} \int \cdots \int f\left(\frac{(x_1 - \mu) + \cdots + (x_n - \mu)}{\sqrt{n}\sigma}\right) p(x_1) \cdots p(x_n) dx_1 \cdots dx_n &= \int_{-\infty}^{\infty} f(z) \frac{e^{-z^2/2}}{\sqrt{2\pi}} dz. \end{aligned}$$

ここで \int は適切な範囲の定積分を意味し, 各 n ごとに n 重の積分を考えている.

このような n 重積分の $n \rightarrow \infty$ での極限公式は非常に複雑な形に見えるが, 大数の法則と中心極限定理を理解していれば直観的に当然そうなるべき結果だと理解できる.

4.5 再掲: 大数の法則と中心極限定理のイメージ

以下の図は

- 「大数の法則と中心極限定理」に関するノート
(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/05%20Central%20limit%20theorem.ipynb>)

にあった図の再掲である.

確率統計に関する入門的教科書の中には, 大数の法則や中心極限定理について説明するために, 確率変数や確率分布に関する様々な種類の収束性の定義を説明しているものが多い. (確率収束, 概収束, 法収束(法則収束, 分布収束)など. 他にも色々ある.)

この一連のノート群においては, そのような説明を意図的に避けた.

なぜならば, 統計学を実践的に使うことが目標の人達は以下の図のイメージを大事した方がよいように思われるからである.

数学的に正確な理解を目指す場合であっても, 最初のうちは以下の図のようなイメージを大事にして, むしろこのイメージを元に, 確率変数や確率分布の収束に関する数学的に適切な定義について, 自分で考えるようにした方が良いように思われる.

教科書に書いてある天下り的な定義から出発しようとすると数学の理解は苦しくなることが多い.

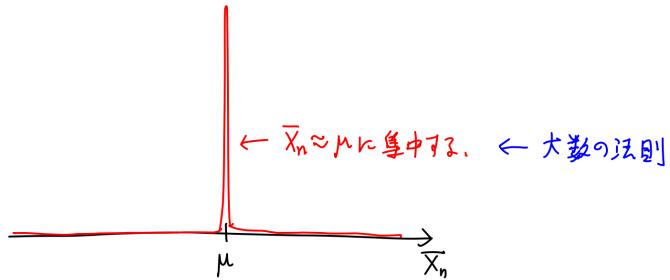
自分の力で適切な定義を作るために十分な「論理的スキル」と「健全な直観」の組み合わせの習得を目指す方が楽に理解できる可能性が増えるだろう.

大数の法則と中心極限定理のイメージ

$$\leftarrow \mu = E[X_i], \sigma = \sqrt{E[(X_i - \mu)^2]} \text{ とおく。}$$

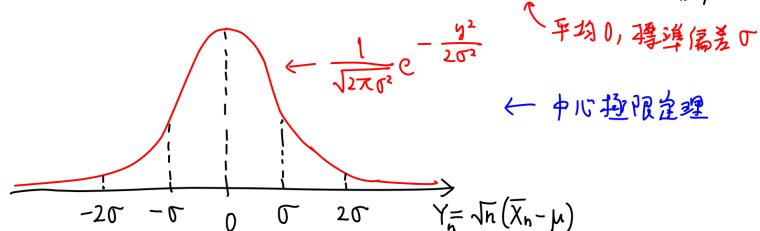
X_1, X_2, X_3, \dots は独立同分布確率変数列であるとし、 $\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$ とおく。

サイスルの標本の平均 \bar{X}_n の分布の $n \rightarrow \infty$ での様子：



$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$ の分布は $n \rightarrow \infty$ で $\mu = E[X_i]$ に集中する。

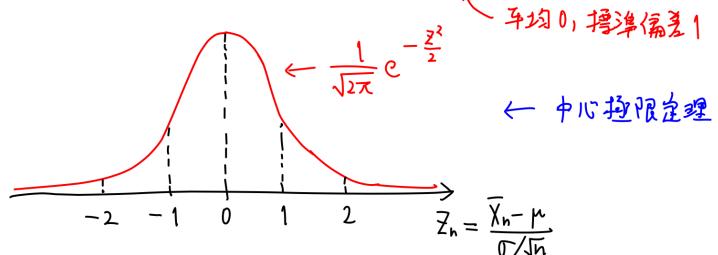
\bar{X}_n と μ の差を \sqrt{n} 倍して拡大して見るとこうなる： $Y_n = \sqrt{n}(\bar{X}_n - \mu) = \frac{1}{\sqrt{n}} \sum_{i=1}^n (X_i - \mu)$,



$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$ と μ の差の大きさはだいたい $\frac{\sigma}{\sqrt{n}}$ の大きさであり、

差を \sqrt{n} 倍拡大して分布を見ると標準偏差 σ の正規分布に近づいている。

さらに σ 分の1倍するところ： $Z_n = \frac{\sqrt{n}}{\sigma} (\bar{X}_n - \mu) = \frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} = \frac{1}{\sqrt{n}} \sum_{i=1}^n \frac{X_i - \mu}{\sigma}$



以上のように、大数の法則によって1点に集中する標本平均の分布を \sqrt{n} 倍 (もしくは σ/\sqrt{n} 分の1倍)することにより精密に見ると中心極限定理が得られる。

4.6 統計学の基礎になる確率論の三種の神器

これは筆者の個人的な意見に過ぎないのだが、統計学の基礎になる確率論の三種の神器は次の3つである：

- 大数の法則
- 中心極限定理
- Kullback-Leibler情報量に関するSanovの定理

この一連のノート群で3つ目の Kullback-Leibler情報量に関するSanovの定理 にはほとんど触れることができなかった。

Kullback-Leibler情報量のSanovの定理については

- 「Bernoulli試行と関連確率分布」に関するノート
(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/01%20Bernoulli%20trial%20and%20related%20distribution.ipynb>)

の「問題: Kullback-Leibler情報量とGibbsの情報不等式」と

- ・「[大数の法則と中心極限定理](https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/05%20Central%20limit%20theorem.ipynb)」に関するノート
(<https://nbviewer.org/github/genkuroki/Statistics/blob/master/2022/05%20Central%20limit%20theorem.ipynb>)

の「注意: Kullback-Leibler情報量とSanovの定理との関係」で簡単に触れただけで終わった。

Kullback-Leibler情報量のSanovの定理は**大偏差原理** (large deviation principle)の特別な場合になっているので、三種の神器の3番目は「大偏差原理」に一般化しておいてもよい。大偏差原理は統計力学的な意味でのエントロピーや除法理論における情報量の概念を扱う話だと思ってよい。

Kullback-Leibler情報量のSanovの定理は赤池情報量規準(AIC)などの情報量規準の基礎になる。

Sanovの定理の易しい解説が以下の場所にある:

- ・[Kullback-Leibler情報量とSanovの定理](https://genkuroki.github.io/documents/20160616KullbackLeibler.pdf) (<https://genkuroki.github.io/documents/20160616KullbackLeibler.pdf>)

確率論で役に立つ解析学については次のノートも参考になるかもしれない:

- ・[ガンマ分布の中心極限定理とStirlingの公式](https://genkuroki.github.io/documents/20160501StirlingFormula.pdf) (<https://genkuroki.github.io/documents/20160501StirlingFormula.pdf>)

ついでに紹介しておくと、ガンマ函数やベータ函数などについては次の場所にあるノート群が詳しい:

- ・[「微分積分学」に関するノート群](https://github.com/genkuroki/Calculus) (<https://github.com/genkuroki/Calculus>)

さらに、図書館で次の文献も参照できれば、統計学の基礎になる確率論の三種の神器について理解を深め易いと思われる:

- ・高橋陽一郎、確率論の広がり、数学のたのしみ、no.8 (1998) pp.26-35 ([関連情報を検索](https://www.google.com/search?q=%E9%AB%98%E6%A9%8B%E9%99%BD%E4%B8%80%E9%83%8E+%E7%A2%BA%E7%8E%87%E8%AB%96%E3%) (<https://www.google.com/search?q=%E9%AB%98%E6%A9%8B%E9%99%BD%E4%B8%80%E9%83%8E+%E7%A2%BA%E7%8E%87%E8%AB%96%E3%>)

面白いことはたくさんある(ありすぎる)。

4.7 Kullback-Leibler情報量に関するSanovの定理の数値例

二項分布の場合のSanovの定理の1つの形: $n \rightarrow \infty, k \sim np$ のとき

$$\lim_{n \rightarrow \infty} \frac{-1}{n} \log \left(\binom{n}{k} q^k (1-q)^{n-k} \right) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}.$$

この公式の右辺はKullback-Leibler情報量の最も簡単な場合になっている。これは

$$\begin{aligned} & (\text{二項分布 } \text{Binomial}(n, q) \text{ で } np \text{ に近い値 } k \text{ が生成される確率}) \\ &= \exp \left(-n \left(p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q} \right) + o(n) \right). \end{aligned}$$

と書き直される。これは、二項分布 $\text{Binomial}(n, q)$ で np に近い値 k が生成される確率の大きさがほぼKullback-Leibler情報量で決まっていることを意味している。

注意: 上の結果は、階乗のStirlingの近似公式を使うと簡単に証明できるし、高校数学で習ったはずに区分求積法を使っても容易に証明できる。注意終

上の結果を数値的に確認してみよう。

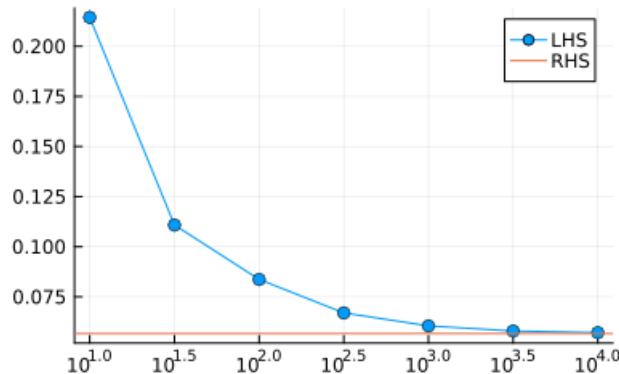
```
In [67]: 1 n, p, q = 10^16, 1/3, 1/2
2 @show LHS = (-1/n)*logpdf(Binomial(n, q), round(Int, n*p))
3 @show RHS = p*log(p/q) + (1-p)*log((1-p)/(1-q));
```

```
LHS = (-1 / n) * logpdf(Binomial(n, q), round(Int, n * p)) = 0.05663301226513299
RHS = p * log(p / q) + (1 - p) * log((1 - p) / (1 - q)) = 0.056633012265132565
```

In [68]:

```
1 ns, p, q = 10 .^ (1:0.5:4), 1/3, 1/2
2 LHSs = [(-1/n)*logpdf(Binomial(round(Int, n), q), round(Int, n*p)) for n in ns]
3 RHS = p*log(p/q) + (1-p)*log((1-p)/(1-q))
4 plot(ns, LHSs; xscale=:log10, xticks=ns, marker=:o, label="LHS")
5 hline!([RHS]; label="RHS")
```

Out[68]:



In []:

```
1
```