標本分布について

- 黒木玄

- 2022-04-11～2022-04-18

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)

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

In [1]:
```
ENV["LINES"], ENV["COLUMNS"] = 100, 100
using BenchmarkTools
using Distributions
using Printf
using QuadGK
using Random
Random.seed!(4649373)
using Roots
using SpecialFunctions
using StaticArrays
using StatsBase
using StatsFuns
using StatsPlots
default(fmt = :png, titlefontsize = 10, size = (400, 250))
using SymPy
```

1 標本分布

1.1 同時確率質量函数と同時確率密度函数

確率変数達 $X_1, \dots, X_n$ が同時確率質量函数 $P(x_1, \dots, x_n)$ を持つとは, $P(x_1, \dots, x_n) \geq 0$ でかつ, $P(x_1, \dots, x_n)$ の総和が 1 であり,

$$E[f(X_1, \dots, X_n)] = \sum_{x_1} \cdots \sum_{x_n} f(x_1, \dots, x_n) P(x_1, \dots, x_n)$$

が成立することである. このとき, X_i 単独の確率質量函数 $P(x_i)$ は

$$P(x_i) = \sum_{x_1} \cdots \widehat{\sum_{x_i}} \cdots \sum_{x_n} P(x_1, \cdots, x_i, \dots, x_n)$$

になる. ここで, $\widehat{\sum_{x_i}}$ は
sum_{x_i}
を除くという意味である. なぜならば,

$$\begin{aligned} E[f(X_i)] &= \sum_{x_1} \cdots \sum_{x_i} \cdots \sum_{x_n} f(x_i) P(x_1, \cdots, x_i, \dots, x_n) \\ &= \sum_{x_i} f(x_1) \left(\sum_{x_1} \cdots \widehat{\sum_{x_i}} \cdots \sum_{x_n} P(x_1, \cdots, x_i, \dots, x_n) \right). \end{aligned}$$

注意: $i \neq j$ のとき $P(x_i)$ と $P(x_j)$ は一般に異なる確率質量函数になることに注意せよ. 区別を明瞭にするためには $P(x_i)$ を $P_i(x_i)$ のように書くべきであるが, 統計学の世界での慣習に従ってこのような記号法上の省略を行った. これが気に入らない人は自分でノートを作るときに, 区別されるべきものに異なる記号を割り振るようにすればよいだろう. 以下についても同様である.

確率変数達 $X_1, \dots, X_n$ が同時確率密度函数 $p(x_1, \dots, x_n)$ を持つとは, $p(x_1, \dots, x_n) \geq 0$ でかつ, $p(x_1, \dots, x_n)$ の積分が 1 になり,

$$E[f(X_1, \dots, X_n)] = \int \cdots \int f(x_1, \dots, x_n) p(x_1, \dots, x_n) \, dx_1 \cdots dx_n$$

が成立することである. ここで $\int$ は定積分を表す. このとき, X_i 単独の確率密度函数 $p(x_i)$ は

$$p(x_i) = \int \cdots \widehat{\int} \cdots \int p(x_1, \cdots, x_i, \dots, x_n) \, dx_1 \cdots \widehat{dx_i} \cdots dx_n$$

になる. ここで, $\widehat{\int}$, $\widehat{dx_i}$ はそれらを除くことを意味する. なぜならば,

$$\begin{aligned} E[f(X_i)] &= \int \cdots \int \cdots \int f(x_i) p(x_1, \cdots, x_i, \dots, x_n) \, dx_1 \cdots dx_i \cdots dx_n \\ &= \int f(x_i) \left(\int \cdots \widehat{\int} \cdots \int p(x_1, \cdots, x_i, \dots, x_n) \, dx_1 \cdots \widehat{dx_i} \cdots dx_n \right) dx_i. \end{aligned}$$

1.2 確率変数の独立性の定義

確率変数達 $X_1, \dots, X_n$ が与えられており, それらの函数の期待値 $E[f(X_1, \dots, X_n)]$ が定義されているとする. このとき, $X_1, \dots, X_n$ が **独立** (independent)であるとは,

$$E[f_1(X_1) \cdots f_n(X_n)] = E[f_1(X_1)] \cdots E[f_n(X_n)] \tag{1}$$

のように X_i ごと函数達 $f_i(X_i)$ の積の期待値が各々の $f_i(X_i)$ の期待値の積になることだと定める.

注意: 厳密には函数 f_i 達を動かす範囲を確定させる必要があるが, このノートではそのようなことにこだわらずに解説することにする. 厳密な理論の展開のためには測度論的確率論の知識が必要になるが, そのような方向はこのノートの目標とは異なる. 測度論的確率論に興味がある人は別の文献を参照して欲しい. ただし, 測度論的確率論の理解と統計学の理解は別の話題になってしまうことには注意して欲しい. 測度論的確率論と統計学では興味の方向が異なる.

1.2.1 独立な確率変数達の同時確率質量函数

確率変数達 $X_1, \dots, X_n$ が同時確率質量函数 $P(x_1, \dots, x_n)$ を持つとき, X_i の確率質量函数を $P_i(x_i)$ と書くとき, 確率変数達 $X_1, \dots, X_n$ が独立であることと,

$$P(x_1, \dots, x_n) = P_1(x_1) \cdots P_n(x_n) \tag{2}$$

が成立することは同値である. すなわち, $(X_1, \dots, X_n)$ の函数の期待値が次のように表されることと同値である:

$$E[f(X_1, \dots, X_n)] = \sum_{x_1} \cdots \sum_{x_n} f(x_1, \dots, x_n) P_1(x_1) \cdots P_n(x_n)$$

注意: ここでは気分を変えて, $P_i(x_i)$ をずぼらに $P(x_i)$ と書く流儀をやめてみた.

証明: (2)が成立しているならば,

$$\begin{aligned} E[f_1(X_1) \cdots f_n(X_n)] &= \sum_{x_1} \cdots \sum_{x_n} f_1(x_1) \cdots f_n(x_n) P_1(x_1) \cdots P_n(x_n) \\ &= \sum_{x_1} f_1(x_1) P_1(x_1) \cdots \sum_{x_n} f_n(x_n) P_n(x_1) \\ &= E[f_1(X_1)] \cdots E[f_n(X_n)] \end{aligned}$$

と(1)が成立する.

逆に(1)が成立しているならば, $f_i(x_i)$ として $x_i = a_i$ のときにのみ 1 でそれ以外のときに 0 になる函数を取ると,

$$P(a_1, \dots, a_n) = E[f_1(X_1) \cdots f_n(X_n)] = E[f_1(X_1)] \cdots E[f_n(X_n)] = P_1(a_1) \cdots P_n(a_n)$$

なので, (2)が成立する.

1.2.2 独立な確率変数達の同時確率密度函数

確率変数達 $X_1, \dots, X_n$ が同時確率密度函数 $p(x_1, \dots, x_n)$ を持つとき, X_i の確率密度函数を $p_i(x_i)$ と書くとき, 確率変数達 $X_1, \dots, X_n$ が独立であることと,

$$p(x_1, \dots, x_n) = p_1(x_1) \cdots p_n(x_n) \tag{3}$$

が成立することは同値である. すなわち, $(X_1, \dots, X_n)$ の函数の期待値が次のように表されることと同値である:

$$E[f(X_1, \dots, X_n)] = \int \cdots \int f(x_1, \dots, x_n) p_1(x_1) \cdots p_n(x_n) \, dx_1 \cdots dx_n.$$

ここで $\int \cdots dx_i$ は適切な範囲での定積分を表す.

注意: ここでは気分を変えて, $p_i(x_i)$ をずぼらに $p(x_i)$ と書く流儀をやめてみた.

(雑な)証明: (3)が成立しているならば,

$$\begin{aligned} E[f_1(X_1) \cdots f_n(X_n)] &= \int \cdots \int f_1(x_1) \cdots f_n(x_n) p_1(x_1) \cdots p_n(x_n) \, dx_1 \cdots dx_n \\ &= \int f_1(x_1) p_1(x_1) \, dx_1 \cdots \int f_n(x_n) p_n(x_n) \, dx_n \\ &= E[f_1(X_1)] \cdots E[f_n(X_n)] \end{aligned}$$

と(1)が成立する.

簡単のため, 密度函数達 $p(x_1, \dots, p_n)$, $p_i(x_i)$ は連続函数になっていると仮定する(応用上は概ねこれで十分). このとき,

$$f_i(x_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(x_i - a_i)^2}{2\sigma_i^2}\right)$$

とおくと, $\sigma_i \searrow 0$ のとき

$$E[f_i(X_i)] = \int f_i(x_i) p_i(x_i)\, dx_i \to p_i(a_i),$$
$$E[f_1(X_1) \cdots f_n(X_n)] = \int \cdots \int f_1(x_1) \cdots f_n(x_n) p(x_1, \ldots, x_n)\, dx_1 \cdots dx_n \to p(a_1, \ldots, a_n)$$

となるので, 逆に(1)が成立しているならば

$$p(a_1, \ldots, a_n) = p_1(a_1) \cdots p_n(a_n)$$

と(3)が成立する.

確率密度函数と確率質量函数が混ざっている場合も同様である.

1.2.3 独立性に関する大雑把なまとめ

大雑把に誤解を恐れずに言うと, 確率が積になるときに独立だという.

実際の計算では, 積の期待値が期待値の積になるという形式で確率変数の独立性を使う.

In [2]:
```
# 以下で使う図の準備

var"正規分布は分散が小さくなると一点に集中して行く." = plot()
for σ in (1.0, 0.5, 0.25, 0.1)
    plot!(Normal(0, σ); label="σ = $σ")
end
title!("Normal(μ=0, σ)")
;
```

1.2.4 分散を0に近付けたときの正規分布について

X_σ を平均 μ, 分散 σ^2 の正規分布に従う確率変数とすると, 有界な連続函数 $f(x)$ について

$$\lim_{\sigma \searrow 0} E[f(X_\sigma)] = f(\mu)$$

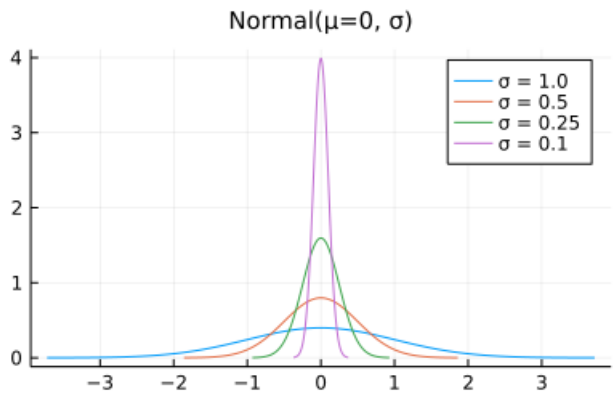
となる. 証明が知りたい人は次のリンク先の「総和核」に関する解説(特にGauss核の説明)を参照せよ:

- 12 Fourier解析 - 総和核 (https://nbviewer.org/github/genkuroki/Calculus/blob/master/12%20Fourier%20analysis.ipynb#%E7%B7%8F%E5%92%8C%E6%

大雑把に言えば, これは $\sigma \searrow 0$ の極限で正規分布が一点 $x = \mu$ に集中することを意味している.

In [3]:
```
var"正規分布は分散が小さくなると一点に集中して行く."
```

Out[3]:



$E[f(X_0)] = f(\mu)$ を満たす確率変数 X_0 は実質定数 μ に等しく, 実質定数に等しい確率変数が従う分布を **Dirac分布** と呼ぶことがある.

1.3 独立同分布の定義

確率変数達 $X_1, \dots, X_n$ が **独立同分布** (independent and identically distributed, **i.i.d.**, **iid**) であるとは, それらが独立でかつ X_i 達が従う分布が互いにすべて等しいことであると定める.

1.4 標本分布の定義

独立同分布な n 個の確率変数達 $X_1, \dots, X_n$ の同時確率分布を各 X_k の分布の **サイズ n の標本分布** (サンプル分布, sample distribution)と呼ぶ.

確率変数 X_i 達が同一の確率密度函数 $p(x_i)$ を持つとき, $X_1, \dots, X_n$ の同時確率密度函数が

$$p(x_1, \dots, x_n) = p(x_1) \cdots p(x_n)$$

であることと, $X_1, \dots, X_n$ の同時確率分布が確率密度函数 $p(x)$ が定める連続分布の標本分布であることは同値である.

確率変数 X_i 達が同一の確率質量函数 $P(x_i)$ を持つとき, $X_1, \dots, X_n$ の同時確率質量函数が

$$P(x_1, \dots, x_n) = P(x_1) \cdots P(x_n)$$

であることと, $X_1, \dots, X_n$ の同時確率分布が確率質量函数 $P(x)$ が定める離散分布の標本分布であることは同値である.

1.5 試行回数 n のBernoulli試行の分布はBernoulli分布の標本分布

試行回数 n, 成功確率 p のBernoulli試行の確率質量函数は

$$P(x_1, \dots, x_n) = p^{x_1+\cdots+x_n}(1-p)^{n-(x_1+\cdots+x_n)} = \prod_{i=1}^{n}(p^{x_i}(1-p)^{1-x_i}) \quad (x_i = 1, 0)$$

であった. これは成功確率 p のBernoulli分布の確率質量函数

$$P(x_i) = p^{x_i}(1-p)^{1-x_i} \quad (x_i = 1, 0)$$

の積になっているので, 試行回数 n, 成功確率 p のBernoulli試行の確率分布は, 成功確率 p のBernoulli分布のサイズ n の標本分布になっている.

未知の確率 p で当たりが出るルーレットを n 回まわしたときの, 長さ n の当たりと外れからなる列をそのルーレットの出目のサイズ n の **標本** (**サンプル**, sample)と呼ぶ.

その意味でのルーレットの出目のサンプルの確率的揺らぎは, Bernoulli分布の標本分布(すなわちBernoulli試行の分布)でモデル化される.

例えば, 未知の確率 p で当たりが出るルーレットを $n = 1000$ 回まわしてサンプル(データ)を取得したら, 1000 回中当たりが 308 回で外れが 692 回ならば, そのルーレットで当たりが出る確率は3割程度だろうと推定できる. 実際にはルーレットを 1000 回まわし直すたびに当たりの回数は確率的に揺らぐので, 推定結果も確率的に揺らぐことになる. そのような揺らぎを数学的にモデル化するために標本分布は使用される.

言葉使いに関する重要な注意: 「標本」「サンプル」は一般に複数の数値の集まりになる. 上のルーレットの場合には当たりには 1 を対応させ, 外れを 0 に対応させると, サンプルは 1 と 0 からなる長さ n の列になる. 一つひとつの数値をサンプルと呼ぶのではなく, 複数の数値の集まりをサンプルと呼ぶ. この専門用語的言葉遣いは日常用語的なサンプルという言葉の使い方からはずれているので注意が必要である. この辺の言葉遣いで誤解を防ぎたい場合には「データ」と呼ぶこともある.

1.6 二項分布による推定の確率的揺らぎの記述

前節の設定を引き継ぐ.

Bernoulli分布のサイズ n の標本分布における 1 の個数の分布は二項分布になるのであった. 仮に n 回中 k 回の当たりが出たときに, 未知の当たりの確率は k/n であると推定することにしたときの, k/n の確率的な揺らぎは二項分布によって計算できる.

K は二項分布 $\mathrm{Binomial}(n, p)$ に従う確率変数であるとし, 確率変数 $\hat{p}$ を $\hat{p} = K/n$ と定める. この確率変数 $\hat{p} = K/n$ は上のルールで定めた未知の確率 p の推定の仕方のモデル化になっている.

$\hat{p} = K/n$ の期待値と分散は, 二項分布の期待値と分散がそれぞれ np と $np(1-p)$ であることより, 以下のように計算される:

$$\begin{aligned} E[\hat{p}] &= E[K/n] = \frac{E[K]}{n} = \frac{np}{n} = p, \\ \mathrm{var}(\hat{p}) &= \mathrm{var}(K/n) = \frac{\mathrm{var}(K)}{n^2} = \frac{np(1-p)}{n^2} = \frac{p(1-p)}{n}. \end{aligned}$$

例えば, $n = 1000$, $p = 0.3$ のとき, $\hat{p}$ の標準偏差は

$$\mathrm{std}(\hat{p}) = \sqrt{\frac{p(1-p)}{n}} = \sqrt{\frac{0.3 \cdot 0.7}{1000}} \approx 1.45\%.$$

モデル内の設定では大雑把に推定結果はこの2倍の $\pm 3\%$ 程度揺らぐと考えらえる.

未知の p を使わずに $\mathrm{SE} = \mathrm{std}(\hat{p})$ の値を推定するためにはここで使った p に $\hat{p}$ を代入して得られる公式

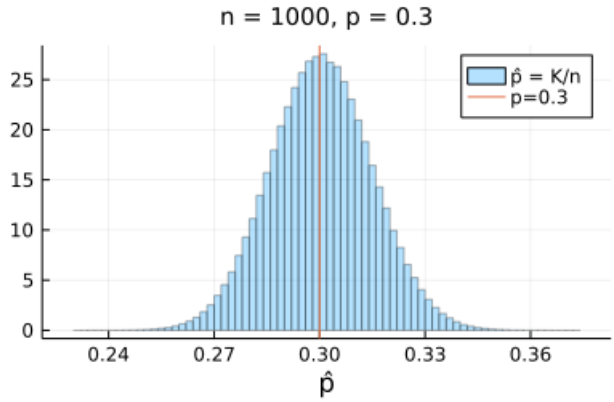
$$\widehat{SE} = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

を使えばよいだろう.

このような統計分析の結果が現実において信頼できるかどうかは, Bernoulli分布の標本分布によるモデル化の現実における妥当性に依存する. モデルが現実において妥当である証拠が全然無ければ, このような推測結果も信頼できないことになる. モデルの現実における妥当性の証拠の提示は統計モデルのユーザー側が独自に行う必要がある.

In [4]:
```
n, p = 1000, 0.3
dist = Binomial(n, p)
L = 10^6
K = rand(dist, L)
p̂ = K/n
histogram(p̂; norm=true, alpha=0.3, label="p̂ = K/n", bin=100)
vline!([p]; label="p=$p", xlabel="p̂", xtick=0.30-0.12:0.03:0.30+0.12)
title!("n = $n, p = $p")
```

Out[4]:



確かにランダムに決まる $\hat{p} = K/n$ の値は, 推定先の値である $p = 0.30$ を中心に大雑把に $\pm 3\%$ 程度の範囲に分布している.

実際には小さな確率でもっと大きく外れることもあることに注意せよ.

現実の統計分析ではこのような未知の確率 $p = 0.3$ に当たるものが使えないので, このようなグラフを描くことはできない. 真っ暗闇の中を手探りで進むような感じになる.

1.7 問題: 大阪都構想に関する住民投票の結果について

2015年と2020年の大阪都構想に関する住民投票の結果は

- 2015年: 賛成: 694,844 (49.6%)　反対: 705,585 (50.4%)
- 2020年: 賛成: 675,829 (49.4%)　反対: 692,996 (50.6%)

であった(検索 (https://www.google.com/search?q=%E5%A4%A7%E9%98%AA%E9%83%BD%E6%A7%8B%E6%83%B3++%E4%BD%8F%E6%B0%91%E6%8A%95%E7%A5%A

どちらでも僅差で反対派が勝利した. パーセントの数値を見ると大変な僅差であったようにも見える. この数値に二項分布モデルを適用したらどうなるかを(それが妥当な適用であるかどうかを度外視して)計算するのがこの問題の内容である.

K は二項分布 $\mathrm{Binomial}(n, p)$ に従うと仮定する.

(1) $n = 694844 + 705585 = 1400429$, $p = 0.5$ のとき, 確率 $P(K \leq 694844)$ の2倍を求めよ.

(2) $n = 675829 + 692996 = 1368825$, $p = 0.5$ のとき, 確率 $P(K \leq 675829)$ の2倍を求めよ.

(3) $n = 694844 + 705585 = 1400429$ のとき, 以下を求めよ:

- $P(K \geq 694844) = 2.5\%$ になるようなパラメータ p の値 p_L,
- $P(K \leq 694844) = 2.5\%$ になるようなパラメータ p の値 p_U.

(4) $n = 675829 + 692996 = 1368825$ のとき, 以下を求めよ:

- $P(K \geq 675829) = 2.5\%$ になるようなパラメータ p の値 p_L,
- $P(K \leq 675829) = 2.5\%$ になるようなパラメータ p の値 p_U.

確率やパラメータの数値は有効桁4桁まで求めよ. $0.000\cdots01234$ のように 0 を沢山含む表示は見難いので,

$$1.234\text{E}{-20} = 1.234\text{e}{-20} = 1.234 \times 10^{20} = 0.\underbrace{00000000000000000001}_{20}234$$

の意味で $1.234\text{E}{-20}$ または $1.234\text{e}{-20}$ のように書くこと.

この問題の内容を一般化するだけで **検定** (統計的仮説検定, statistical hypothesis testing)や **信頼区間** (confidence interval)の一般論が得られる.

解答例:

(1) $P(K \leq 694844) \approx 1.130\text{e}{-19}$

(2) $P(K \leq 675829) \approx 9.687\text{e}{-49}$

(3) $[p_L, p_U] \approx [0.4953, 0.4970]$

(4) $[p_L, p_U] \approx [0.4929, 0.4946]$

解答終

1.7.1 Julia言語による計算の例

In [5]:
```
# 確率計算を素朴に行うには対数を取った結果を主な対象にしないと失敗する.
# 次は二項分布における確率質量函数の対数である.
logP(n, p, k) = logabsbinomial(n, k)[1] + k*log(p) + (n-k)*log(1-p)
```

Out[5]:
```
logP (generic function with 1 method)
```

In [6]:
```
# (1)
@show 2exp(logsumexp(logP(694844 + 705585, 0.5, k) for k in 0:694844))
@show 2cdf(Binomial(694844 + 705585, 0.5), 694844);
```

```
2 * exp(logsumexp((logP(694844 + 705585, 0.5, k) for k = 0:694844))) = 1.130422573100686e-19
2 * cdf(Binomial(694844 + 705585, 0.5), 694844) = 1.1304225734350622e-19
```

In [7]:
```
# (2)
@show 2exp(logsumexp(logP(675829 + 692996, 0.5, k) for k in 0:675829))
@show 2cdf(Binomial(675829 + 692996, 0.5), 675829);
```

```
2 * exp(logsumexp((logP(675829 + 692996, 0.5, k) for k = 0:675829))) = 9.687442124914083e-49
2 * cdf(Binomial(675829 + 692996, 0.5), 675829) = 9.687442131513807e-49
```

In [8]:
```
# (3)
f(t) = ccdf(Binomial(694844 + 705585, t), 694844-1) - 0.025
g(t) =  cdf(Binomial(694844 + 705585, t), 694844)   - 0.025
@show p_L = find_zero(f, (0, 1))
@show p_U = find_zero(g, (0, 1));
```

```
p_L = find_zero(f, (0, 1)) = 0.4953366705819138
p_U = find_zero(g, (0, 1)) = 0.4969935526649652
```

In [9]:
```
# (3)
n = 694844 + 705585
k = 694844
α = 0.05
@show p_L = quantile(Beta(k, n-k+1), α/2)
@show p_U = quantile(Beta(k+1, n-k), 1 - α/2);
```

```
p_L = quantile(Beta(k, (n - k) + 1), α / 2) = 0.4953366705819138
p_U = quantile(Beta(k + 1, n - k), 1 - α / 2) = 0.49699355266496525
```

In [10]:
```
# (3)
n = 694844 + 705585
k = 694844
α = 0.05
@show p_L = beta_inc_inv(k, n-k+1, α/2)[1]
@show p_U = beta_inc_inv(k+1, n-k, 1 - α/2)[1];
```

```
p_L = (beta_inc_inv(k, (n - k) + 1, α / 2))[1] = 0.4953366705819138
p_U = (beta_inc_inv(k + 1, n - k, 1 - α / 2))[1] = 0.49699355266496514
```

In [11]:
```
# (4)
f(t) = ccdf(Binomial(675829 + 692996, t), 675829-1) - 0.025
g(t) =  cdf(Binomial(675829 + 692996, t), 675829)   - 0.025
@show p_L = find_zero(f, (0, 1))
@show p_U = find_zero(g, (0, 1));
```

```
p_L = find_zero(f, (0, 1)) = 0.49289139358630474
p_U = find_zero(g, (0, 1)) = 0.4945672196912542
```

In [12]:
```
# (4)
n = 675829 + 692996
k = 675829
α = 0.05
@show p_L = quantile(Beta(k, n-k+1), α/2)
@show p_U = quantile(Beta(k+1, n-k), 1 - α/2);
```

```
p_L = quantile(Beta(k, (n - k) + 1), α / 2) = 0.49289139358630474
p_U = quantile(Beta(k + 1, n - k), 1 - α / 2) = 0.4945672196912543
```

In [13]:
```
# (4)
n = 675829 + 692996
k = 675829
α = 0.05
@show p_L = beta_inc_inv(k, n-k+1, α/2)[1]
@show p_U = beta_inc_inv(k+1, n-k, 1 - α/2)[1];
```

```
p_L = (beta_inc_inv(k, (n - k) + 1, α / 2))[1] = 0.49289139358630474
p_U = (beta_inc_inv(k + 1, n - k, 1 - α / 2))[1] = 0.4945672196912543
```

In [14]:
```
@doc beta_inc_inv
```

Out[14]:
```
beta_inc_inv(a, b, p, q=1-p)
```

Return a tuple `(x, 1-x)` where `x` satisfies $I_x(a,b) = p$, i.e., `x` is the inverse of the regularized incomplete beta function $I_x(a,b)$.

See also: `beta_inc` (@ref)

In [15]:
```
@doc beta_inc
```

Out[15]:
```
beta_inc(a, b, x, y=1-x)
```

Return a tuple $(I_x(a,b), 1 - I_x(a,b))$ where $I_x(a,b)$ is the regularized incomplete beta function given by

$$I_x(a,b) = \frac{1}{B(a,b)} \int_0^x t^{a-1}(1-t)^{b-1}\,dt,$$

where $B(a,b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$.

External links: DLMF (https://dlmf.nist.gov/8.17.1), Wikipedia (https://en.wikipedia.org/wiki/Beta_function#Incomplete_beta_function)

See also: `beta_inc_inv` (@ref)

1.7.2 WolframAlphaによる計算の例:

(1) 2 cdf(BinomialDistribution(694844 + 705585, 0.5), 694844) (https://www.wolframalpha.com/input?i=2+cdf%28BinomialDistribution%28694844+%2B+705585%2C+0.5%29%2C+694844%29)

(2) 2 cdf(BinomialDistribution(675829 + 692996, 0.5), 675829) (https://www.wolframalpha.com/input?i=2+cdf%28BinomialDistribution%28675829+%2B+692996%2C+0.5%29%2C+675829%29)

(3) p_L: solve cdf(BinomialDistribution(694844 + 705585, q), 705585) = 0.025 (https://www.wolframalpha.com/input?i=solve+cdf%28BinomialDistribution%28694844+%2B+705585%2C+q%29%2C+705585%29+%3D+0.025) として, これを1から引いた値を求める: InverseBetaRegularized(1/40, 694844, 705586) (https://www.wolframalpha.com/input?i=InverseBetaRegularized%281%2F40%2C+694844%2C+705586%29). もしくは, InverseBetaRegularized(0.025, 694844, 705585 + 1) (https://www.wolframalpha.com/input?i=InverseBetaRegularized%281%2F40%2C+694844%2C+705586%29)

p_U: solve cdf(BinomialDistribution(694844 + 705585, p), 694844) = 0.025 (https://www.wolframalpha.com/input?i=InverseBetaRegularized%280.025%2C+694844%2C+705585+%2B+1%29). もしくは, InverseBetaRegularized(1 - 0.025, 694844 + 1, 705585) (https://www.wolframalpha.com/input?i=InverseBetaRegularized%281+-+0.025%2C+694844+%2B+1%2C+705585%29&lang=ja)

(4) p_L: solve cdf(BinomialDistribution(675829 + 692996, q), 692996) = 0.025 (https://www.wolframalpha.com/input?i=solve+cdf%28BinomialDistribution%28694844+%2B+705585%2C+q%29%2C+705585%29+%3D+0.025) として, これを1から引いた値を求める: InverseBetaRegularized(1/40, 694844, 705586) (https://www.wolframalpha.com/input?i=InverseBetaRegularized%281%2F40%2C+694844%2C+705586%29). もしくは, InverseBetaRegularized(0.025, 675829, 692996 + 1) (https://www.wolframalpha.com/input?i=InverseBetaRegularized%280.025%2C+675829%2C+692996+%2B+1%29)

p_U: solve cdf(BinomialDistribution(675829 + 692996, p), 675829) = 0.025 (https://www.wolframalpha.com/input?i=solve+cdf%28BinomialDistribution%28675829+%2B+692996%2C+p%29%2C+675829%29+%3D+0.025). もしくは, InverseBetaRegularized(1 - 0.025, 675829 + 1, 692996) (https://www.wolframalpha.com/input?i=InverseBetaRegularized%281+-+0.025%2C+675829+%2B+1%2C+692996%29)

ここで, (3), (4) の p_L の計算では $\mathrm{Binomial}(n, p)$ で k 以上の確率は $\mathrm{Binomial}(n, 1-p)$ で $n-k$ 以下になる確率に等しいことを使った.

1.7.3 Clopper-Pearsonの信頼区間とそれを与えるP値

(3)と(4)で計算した値から得られる区間 $[p_L, p_U]$ は **母比率に関する信頼度95%のClopper-Pearsonの信頼区間** として統計学ユーザーのあいだでよく知られている.

(1)と(2)での2倍する前の確率は **片側検定のP値** の一種である. 2倍後の値は **両側検定のP値** (通常はこちらを使う)の一種になっており, Clopper-Pearsonの信頼区間を与える.

P値は採用した統計モデルとデータの整合性の指標である(P値が小さければ整合性が低い). (1)と(2)で求めたP値の値は極めて小さいということは, 成功確率 $p = 0.5$ の二項分布モデルと2015年と2020年の大阪都構想に関する住民投票の結果の整合性が極めて低いということを意味している. 2015年と2020年の大阪都構想に関する住民投票の結果については, 成功確率 $p = 0.5$ の二項分布モデルは捨て去る必要がある.

(3)と(4)で求めた区間 $[p_L, p_U]$ はデータから計算されるP値が 5% 以上になるパラメータ p の値全体の集合になっている. すなわち, P値に関する 5% の閾値(**有意水準** と呼ばれる)で整合性が低すぎるという理由で捨て去られずにすむ p の値全体が信頼度 $1 - 5\% = 95\%$ の信頼区間になっている. このパターンは一般の場合にもそのまま通用する.

Clopper-Pearsonの信頼区間の効率的計算には, 二項分布の累積分布函数はベータ分布の累積分布函数で書けることが使われる:

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

これらの公式から, n, k が与えられていて $K \sim \mathrm{Binomial}(n, p)$ のとき, $P(K \geq k) = \alpha/2$ の解 p_L と $P(K \leq k) = \alpha/2$ の解 p_U はそれぞれ次のように書ける:

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

ベータ分布の累積分布函数が **正則化された不完全ベータ函数** (regularized incomplete Beta function)になっている:

$$P(T \leq \theta) = I_\theta(\alpha, \beta) = \frac{\int_0^\theta t^{\alpha-1}(1-t)^{\beta-1}}{B(\alpha, \beta)} \quad \text{if} \quad T \sim \mathrm{Beta}(\alpha, \beta)$$

このことから, `quantile` 函数の代わりに $\theta \mapsto p = I_\theta(\alpha, \beta)$ の逆函数を直接使ってもよい:

$$p_L = \mathrm{beta\_inc\_inv}(k, n-k+1, \alpha/2)[1], \quad p_U = \mathrm{beta\_inc\_inv}(k+1, n-k, 1-\alpha/2)[1].$$

Julia言語の `SpecialFunctions.jl` では正則化された不完全ベータ函数とその逆函数はそれぞれ `using SpecialFunctions` した後に `p = beta_inc(α, β, θ)[1]` と `θ = beta_inc_inv(α, β, p)` で使える.

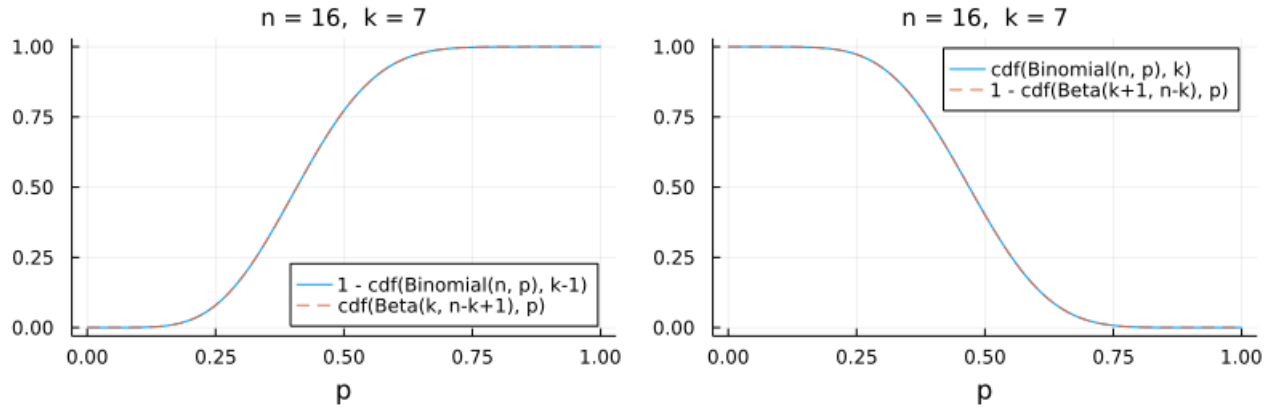
Wolfram言語では `BetaRegularized` (https://reference.wolfram.com/language/ref/BetaRegularized.html) と InverseBetaRegularized (https://reference.wolfram.com/language/ref/InverseBetaRegularized.html) を使う.

Clopper-Pearsonの信頼区間を使うことのメリットは, 二項分布の累積分布函数を和で計算してからパラメータに関する方程式を解くという面倒な手続きを経由せずに, 基本特殊函数の1つである正則化された不完全ベータ函数の逆函数に帰着して効率的に計算できることである. 別の信頼区間の定義の仕方との比較でデメリットもある. Sternの信頼区間(Stern (1954)

(https://www.jstor.org/stable/2333026))との相対比較では, Clopper-Pearsonの信頼区間の方が無駄に広くなってしまう場合が多い. np が大きな場合にはどちらを使っても実践的に意味のある差は出ない.

In [16]:
```
n, k = 16, 7

P1 = plot(p -> 1 - cdf(Binomial(n, p), k-1), 0, 1;
    label="1 - cdf(Binomial(n, p), k-1)", legend=:bottomright)
plot!(p -> cdf(Beta(k, n-k+1), p), 0, 1;
    label="cdf(Beta(k, n-k+1), p)", ls=:dash)
title!("n = $n,  k = $k"; xlabel="p")

P2 = plot(p -> cdf(Binomial(n, p), k), 0, 1;
    label="cdf(Binomial(n, p), k)", legend=:topright)
plot!(p -> 1 - cdf(Beta(k+1, n-k), p), 0, 1;
    label="1 - cdf(Beta(k+1, n-k), p)", ls=:dash)
title!("n = $n,  k = $k"; xlabel="p")

plot(P1, P2; size = (800, 250), bottommargin=4Plots.mm)
```

Out[16]:



1.7.4 信頼区間よりも情報量が大きなP値函数のプロット

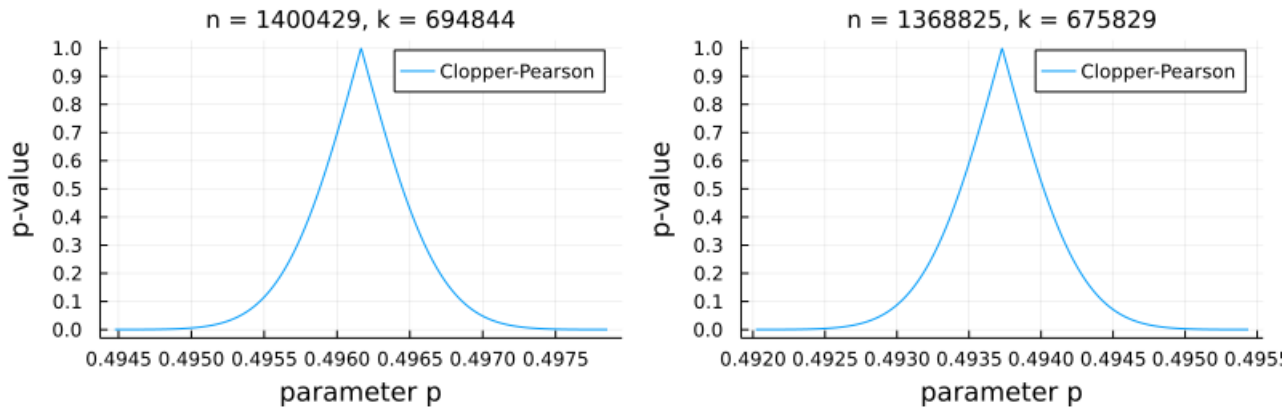
二項分布モデルにおいて n, k が与えられたときに, パラメータ p に対してP値を対応させる函数を **P値函数** (p-value function)と呼ぶ. P値函数の値が有意水準 α 以上の p 全体の集合が信頼度 $1-\alpha$ の信頼区間になる. この意味でP値函数はすべての信頼度に関する信頼区間の情報をすべて持っており, 適当な条件の下ではすべての信頼度に関する信頼区間が与えられていればそこからP値函数を逆に作れる. この意味でP値函数と信頼区間達は表裏一体の関係になっている.

以下では上の問題の場合についてのP値函数をプロットしてみよう.

In [17]:

```
function plot_pvalue_function!(pvalue_func, n, k; label="", kwargs...)
    p̂ = k/n
    SE = √(p̂*(1 - p̂)/n)
    ps = range(p̂ - 4SE, p̂ + 4SE, 1000)
    plot!(ps, p -> pvalue_func(n, k, p); label, kwargs...)
end

function plot_pvalue_function(pvalue_func, n, k; label="", kwargs...)
    plot()
    plot_pvalue_function!(pvalue_func, n, k; label)
    title!("n = $n, k = $k")
    plot!(; ytick=0:0.1:1, xlabel="parameter p", ylabel="p-value")
    plot!(; kwargs...)
end

function pvalue_clopper_pearson(dist::DiscreteUnivariateDistribution, x)
    min(1, 2cdf(dist, x), 2ccdf(dist, x-1))
end
pvalue_clopper_pearson(n, k, p) = pvalue_clopper_pearson(Binomial(n, p), k)

# (3)
P1 = plot_pvalue_function(pvalue_clopper_pearson, 694844 + 705585, 694844;
    label="Clopper-Pearson", xtick=0:0.0005:1)

# (4)
P2 = plot_pvalue_function(pvalue_clopper_pearson, 675829 + 692996, 675829;
    label="Clopper-Pearson", xtick=0:0.0005:1)

plot(P1, P2; size=(800, 250), leftmargin=4Plots.mm, bottommargin=4Plots.mm)
```

Out[17]:



1.7.5 Sternの信頼区間とそれを与えるP値函数

Sternの信頼区間を与えるP値函数の定義は, 二項分布の確率質量函数を

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

と書くとき,

$$\mathrm{pvalue}_{\mathrm{Stern}}(k|n,p) = \sum_{P(j|n,p) \le P(k|n,p)} P(j|n,p)$$

と $P(k|n,p)$ 以下となるような $P(j|n,p)$ 達の和として定義される. すなわち, 二項分布 $\mathrm{Binomial}(n,p)$ においてその値が生じる確率がデータの数値 k が生じる確率以下になる確率としてSternの信頼区間を与えるP値は定義される.

そして, 与えられた n, k について, Sternの信頼区間はこのP値が α 以上になるパラメータ p の範囲として定義される. (実はその定義だと区間になるとは限らない場合があるので, 適当に定義を訂正することになる.)

1.7.6 Sternの信頼区間を与えるP値函数の実装例

以下はSternの信頼区間を与えるP値函数の実装例である.

Clopper-Pearsonの信頼区間を与えるP値函数の実装(実質1行!)と比較すると相当に複雑になっている.

そして, 実装の仕方によって計算効率に大きな違いが生じていることにも注目せよ.

In [18]:

```
x ≲ y = x < y || x ≈ y

# Naive implementation is terribly slow.
function pvalue_stern_naive(dist::DiscreteUnivariateDistribution, x; xmax = 10^6)
    Px = pdf(dist, x)
    Px == 0 && return Px
    ymin, maxdist = minimum(dist), maximum(dist)
    ymax = maxdist == Inf ? xmax : maxdist
    sum(pdf(dist, y) for y in ymin:ymax if 0 < pdf(dist, y) ≲ Px; init = 0.0)
end
pvalue_stern_naive(n, k, p) = pvalue_stern_naive(Binomial(n, p), k)

# Second implementation is very slow.
function pvalue_stern_old(dist::DiscreteUnivariateDistribution, x)
    Px = pdf(dist, x)
    Px == 0 && return Px
    distmin, distmax = extrema(dist)
    m = mode(dist)
    Px ≈ pdf(dist, m) && return one(Px)
    if x < m
        y = m + 1
        while !(pdf(dist, y) ≲ Px)
            y += 1
        end
        cdf(dist, x) + ccdf(dist, y-1)
    else # k > m
        y = m - 1
        while !(pdf(dist, y) ≲ Px)
            y -= 1
        end
        cdf(dist, y) + ccdf(dist, x-1)
    end
end
pvalue_stern_old(n, k, p) = pvalue_stern_old(Binomial(n, p), k)

### efficient implementation

_pdf_le(x, (dist, y)) =  pdf(dist, x) ≲ y

function _search_boundary(f, x0, Δx, param)
    x = x0
    if f(x, param)
        while f(x - Δx, param) x -= Δx end
    else
        x += Δx
        while !f(x, param) x += Δx end
    end
    x
end

function pvalue_stern(dist::DiscreteUnivariateDistribution, x)
    Px = pdf(dist, x)
    Px == 0 && return Px
    m = mode(dist)
    Px ≈ pdf(dist, m) && return one(Px)
    if x < m
        y = _search_boundary(_pdf_le, 2m - x, 1, (dist, Px))
        cdf(dist, x) + ccdf(dist, y-1)
    else # x > m
        y = _search_boundary(_pdf_le, 2m - x, -1, (dist, Px))
        cdf(dist, y) + ccdf(dist, x-1)
    end
end
pvalue_stern(n, k, p) = pvalue_stern(Binomial(n, p), k)
```

Out[18]: pvalue_stern (generic function with 2 methods)

In [19]:
```
n = 10
k = -1:11
p = 0.4
a = @time pvalue_stern_naive.(n, k, p)
b = @time pvalue_stern_old.(n, k, p)
c = @time pvalue_stern.(n, k, p)
d = @time pvalue_clopper_pearson.(n, k, p)
@show a ≈ b ≈ c
[a b c d]
```

```
  0.097843 seconds (440.70 k allocations: 24.314 MiB, 99.64% compilation time)
  0.063668 seconds (211.57 k allocations: 11.444 MiB, 99.41% compilation time)
  0.099706 seconds (190.21 k allocations: 10.235 MiB, 24.21% gc time, 99.66% compilation time)
  0.030186 seconds (125.54 k allocations: 6.829 MiB, 98.82% compilation time)
a ≈ b ≈ c = true
```

Out[19]:
```
13×4 Matrix{Float64}:
 0.0          0.0          0.0          0.0
 0.00772434   0.00772434   0.00772434   0.0120932
 0.058652     0.058652     0.058652     0.0927148
 0.333528     0.333528     0.333528     0.33458
 0.749177     0.749177     0.749177     0.764561
 1.0          1.0          1.0          1.0
 0.534186     0.534186     0.534186     0.733793
 0.212596     0.212596     0.212596     0.332477
 0.101119     0.101119     0.101119     0.109524
 0.0183412    0.0183412    0.0183412    0.0245891
 0.00167772   0.00167772   0.00167772   0.00335544
 0.000104858  0.000104858  0.000104858  0.000209715
 0.0          0.0          0.0          0.0
```

In [20]:
```
# (3)の場合に
# pvalue_stern_naive は pvalue_stern_old よりも数百倍遅く,
# pvalue_stern_old は pvalue_stern よりも数百倍遅く,
# pvalue_stern は pvalue_clopper_pearson よりも少し遅い.
n = 694844 + 705585
k = 694844
a = @btime pvalue_stern_naive($n, $k, 0.5)
b = @btime pvalue_stern_old($n, $k, 0.5)
c = @btime pvalue_stern($n, $k, 0.5)
d = @btime pvalue_clopper_pearson($n, $k, 0.5)
@show a ≈ b ≈ c ≈ d
a, b, c, d
```

```
  136.733 ms (0 allocations: 0 bytes)
  562.200 μs (0 allocations: 0 bytes)
  1.580 μs (0 allocations: 0 bytes)
  1.110 μs (0 allocations: 0 bytes)
a ≈ b ≈ c ≈ d = true
```

Out[20]:
```
(1.1304225731963725e-19, 1.1304225734350622e-19, 1.1304225734350622e-19, 1.1304225734350622e-19)
```

In [21]:
```
# 極端な場合
n = 694844 + 705585
k = 600000
b = @btime pvalue_stern_old($n, $k, 0.5)
c = @btime pvalue_stern($n, $k, 0.5)
d = @btime pvalue_clopper_pearson($n, $k, 0.5)
b, c, d
```

```
  102.682 ns (0 allocations: 0 bytes)
  102.447 ns (0 allocations: 0 bytes)
  326.606 ns (0 allocations: 0 bytes)
```

Out[21]:
```
(0.0, 0.0, 0.0)
```

In [22]:
```
# この場合には pvalue_stern_naive はさらに遅い.
n = 100000
k = 49500:50500
a = @time pvalue_stern_naive.(n, k, 0.5)
b = @time pvalue_stern_old.(n, k, 0.5)
c = @time pvalue_stern.(n, k, 0.5)
d = @time pvalue_clopper_pearson.(n, k, 0.5)
@show a ≈ b ≈ c ≈ d;
```

```
 11.331145 seconds (3 allocations: 8.094 KiB)
  0.027464 seconds (3 allocations: 8.094 KiB)
  0.001485 seconds (3 allocations: 8.094 KiB)
  0.001244 seconds (3 allocations: 8.094 KiB)
a ≈ b ≈ c ≈ d = true
```

In [23]:
```
# 以上の実装は超幾何分布でも使える.
dist = Hypergeometric(9, 9, 9)
k = -1:10
a = @time pvalue_stern_naive.(dist, k)
b = @time pvalue_stern_old.(dist, k)
c = @time pvalue_stern.(dist, k)
d = @time pvalue_clopper_pearson.(dist, k)
@show a ≈ b ≈ c ≈ d
[a b c d]
```

```
  0.081853 seconds (360.44 k allocations: 19.761 MiB, 13.14% gc time, 99.60% compilation time)
  0.047461 seconds (166.44 k allocations: 8.989 MiB, 97.35% compilation time)
  0.049205 seconds (150.20 k allocations: 8.065 MiB, 99.33% compilation time)
  0.034874 seconds (171.76 k allocations: 8.862 MiB, 99.10% compilation time)
a ≈ b ≈ c ≈ d = true
```

Out[23]:
```
12×4 Matrix{Float64}:
 0.0         0.0         0.0         0.0
 4.11353e-5  4.11353e-5  4.11353e-5  4.11353e-5
 0.0033731   0.0033731   0.0033731   0.0033731
 0.0566845   0.0566845   0.0566845   0.0566845
 0.346935    0.346935    0.346935    0.346935
 1.0         1.0         1.0         1.0
 1.0         1.0         1.0         1.0
 0.346935    0.346935    0.346935    0.346935
 0.0566845   0.0566845   0.0566845   0.0566845
 0.0033731   0.0033731   0.0033731   0.0033731
 4.11353e-5  4.11353e-5  4.11353e-5  4.11353e-5
 0.0         0.0         0.0         0.0
```

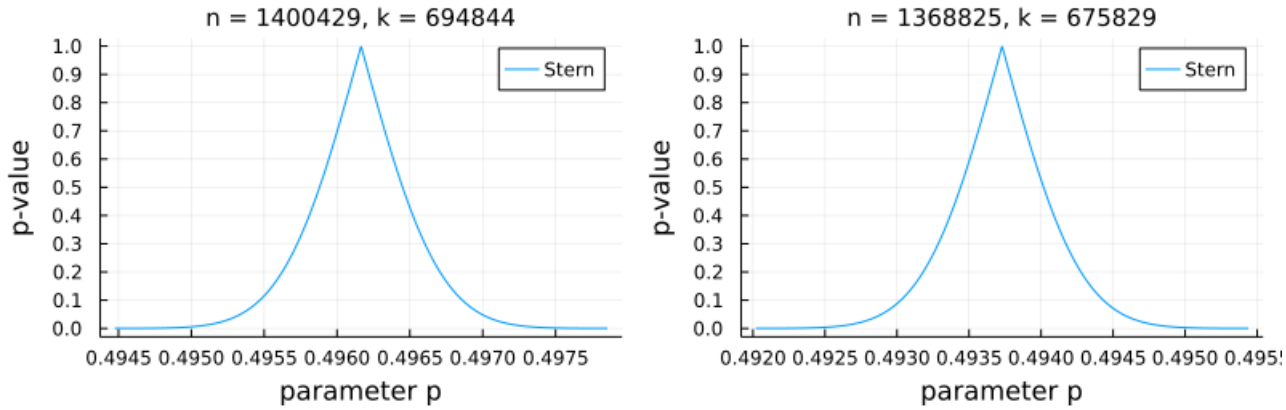
In [24]:
```
# 以上の実装はPoisson分布でも使える.
dist = Poisson(4)
k = -1:10
a = @time pvalue_stern_naive.(dist, k)
b = @time pvalue_stern_old.(dist, k)
c = @time pvalue_stern.(dist, k)
d = @time pvalue_clopper_pearson.(dist, k)
@show a ≈ b ≈ c
[a b c d]
```

```
  0.574708 seconds (433.82 k allocations: 24.206 MiB, 16.15% compilation time)
  0.068191 seconds (158.26 k allocations: 8.643 MiB, 19.62% gc time, 99.40% compilation time)
  0.053509 seconds (145.64 k allocations: 7.904 MiB, 99.25% compilation time)
  0.041185 seconds (165.00 k allocations: 8.612 MiB, 99.02% compilation time)
a ≈ b ≈ c = true
```

Out[24]:
```
12×4 Matrix{Float64}:
 0.0         0.0         0.0         0.0
 0.0396791   0.0396791   0.0396791   0.0366313
 0.202252    0.202252    0.202252    0.183156
 0.452973    0.452973    0.452973    0.476207
 1.0         1.0         1.0         0.86694
 1.0         1.0         1.0         1.0
 0.609266    0.609266    0.609266    0.742326
 0.306448    0.306448    0.306448    0.429739
 0.12899     0.12899     0.12899     0.221348
 0.0694493   0.0694493   0.0694493   0.102267
 0.0213634   0.0213634   0.0213634   0.0427269
 0.00813224  0.00813224  0.00813224  0.0162645
```

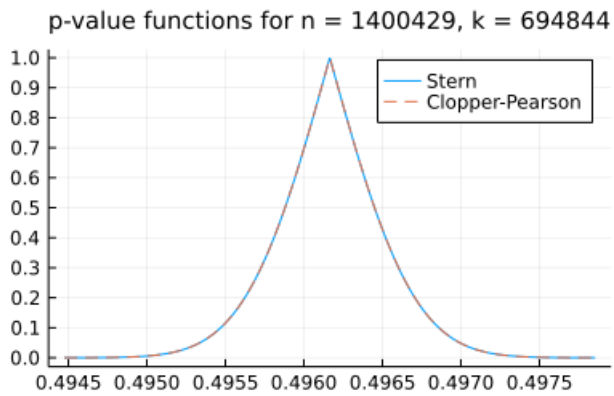
In [25]:
```
# (3)
P1 = plot_pvalue_function(pvalue_stern, 694844 + 705585, 694844;
    label="Stern", xtick=0:0.0005:1)

# (4)
P2 = plot_pvalue_function(pvalue_stern, 675829 + 692996, 675829;
    label="Stern", xtick=0:0.0005:1)

plot(P1, P2; size=(800, 250), leftmargin=4Plots.mm, bottommargin=4Plots.mm)
```

Out[25]:



In [26]:
```
# Clopper-Pearsonの信頼区間を与えるP値函数とSternの信頼区間を与えるP値函数の比較
# (3)の場合にはほぼぴったり一致している.
n, k = 694844 + 705585, 694844
plot(title="p-value functions for n = $n, k = $k", ytick=0:0.1:1)
plot_pvalue_function!(pvalue_stern, n, k; label="Stern")
plot_pvalue_function!(pvalue_clopper_pearson, n, k; label="Clopper-Pearson", ls=:dash)
```

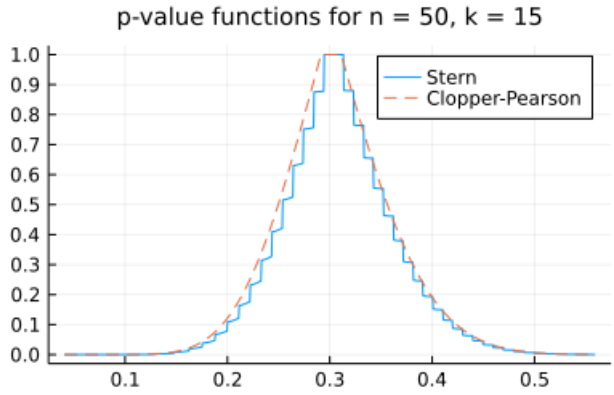
Out[26]:



このように数値的にぴったり一致する場合にはClopper-Pearsonの信頼区間を与えるP値函数とSternの信頼区間のどちらを使うか悩む必要はないだろう. どちらを使っても(ほぼ)同じ結果が得られる.

In [27]:
```
# Clopper-Pearsonの信頼区間を与えるP値函数とSternの信頼区間を与えるP値函数の比較
# n = 50, k = 15 の場合には違いが見える.
n, k = 50, 15
plot(title="p-value functions for n = $n, k = $k", ytick=0:0.1:1)
plot_pvalue_function!(pvalue_stern, n, k; label="Stern")
plot_pvalue_function!(pvalue_clopper_pearson, n, k; label="Clopper-Pearson", ls=:dash)
```

Out[27]:



Sternの信頼区間を与えるP値函数の値はClopper-Pearsonの信頼区間を与えるP値函数の値よりも小さくなりことが多い. (常にそうなるわけではない.) その結果, 対応する信頼区間もSternの側が狭くなってくれることが多い.

1.8 対ごとに独立であっても全体が独立であるとは限らない

確率変数達 X, Y, Z の互いに異なる任意の2つが独立であっても, X, Y, Z の全体が独立でない場合があることを具体的な例を作ることによって証明しよう.

確率変数 X, Y, Z の同時確率質量函数が $P(x, y, z)$ であるとき, X 単独の確率質量函数は $P(x) = \sum_{y,z} P(x, y, z)$ になり, (X, Y) の同時確率質量函数は $P(x, y) = \sum_{z} P(x, y, z)$ になる. 他の場合も同様である.

注意: 変数名で異なる函数を区別するスタイルを採用したので記号法の運用時に注意すること. このスタイルでは $P(x, y)$ と $P(x, z)$ は異な函数になる. 値を代入する場合には $P(x = 1, y = 1)$ や $P(x = 1, z = 1)$ のように書いて区別できるようにする.

$P(x, y, z)$ で $P(x, y) = P(x)P(y)$, $P(x, z) = P(x)P(z)$, $P(y, z) = P(y)P(z)$ を満たすが $P(x, y, z) \neq P(x)P(y)P(z)$ となるものを具体的に構成すればよい.

例えば以下のような確率質量函数 $P(x, y, z)$ の例を作ることができる:

	$z = 1$		$z = 0$	
	$y = 1$	$y = 0$	$y = 1$	$y = 0$
$x = 1$	$P(1, 1, 1) = 1/4$	$P(1, 0, 1) = 0$	$P(1, 1, 0) = 0$	$P(1, 0, 0) = 1/4$
$x = 0$	$P(0, 1, 1) = 0$	$P(0, 0, 1) = 1/4$	$P(0, 1, 0) = 1/4$	$P(0, 0, 0) = 0$

x, y, z はそれぞれ $1, 0$ を動くとする. このとき, $P(x, y)$, $P(x, z)$, $P(y, z)$ の値はすべて $1/4$ になり, $P(x)$, $P(y)$, $P(z)$ の値はすべて $1/2$ になることがわかる. たとえば, $P(x = 1, y = 1) = P(1, 1, 1) + P(1, 1, 0) = 1/4 + 0 = 1/4$ であり, $P(x = 1) = \sum_{y,z} P(1, y, z) = 1/4 + 0 + 0 + 1/4 = 1/2$.

だから, $P(x, y) = P(x)P(y)$, $P(x, z) = P(x)P(z)$, $P(y, z) = P(y)P(z)$ が成立するが, $P(x, y, z) \neq 1/8 = P(x)P(y)P(z)$ となっている.

このとき, X, Y, Z が同時確率質量函数 $P(x, y, z)$ を持つとすると, そのうちの任意の異なる2つは独立だが, X, Y, Z の全体は独立ではない.

1.9 確率変数の独立性の現実における解釈に関する重大な注意

上の例は具体的には次のような状況だと解釈可能である.

(1) $P(x) = 1/2$ の解釈: $X = 1$ は薬Aを与えたことを, $X = 0$ は薬Bを与えたことを意味する. 全員に確率 $1/2$ で薬AまたはBを与えた.

(2) $P(y) = 1/2$ の解釈: $Y = 1$ は薬に効果があったことを, $Y = 0$ は効果が無かったことを意味する. 全体で見たら, $1/2$ の確率で薬には効果があった.

(3) $P(z) = 1/2$ の解釈: $Z = 1$ は女性であることをを, $Z = 0$ は男性であることを意味する. 女性である確率と男性である確率は $1/2$ だった.

(4) 男女の区別をやめると, 薬Aも薬Bも効果がある確率は半々であり, 薬Aと薬Bのどちらを与えたかと効果があったかどうかは独立である. 男女を合わせた($z = 1, 0$ の場合の和を取って得られる)確率質量函数 $P(x, y)$ の表

	$y = 1$	$y = 0$
$x = 1$	$P(x = 1, y = 1) = 1/4$	$P(x = 1, y = 0) = 1/4$
$x = 0$	$P(x = 0, y = 1) = 1/4$	$P(x = 0, y = 0) = 1/4$

は$x = 1$ の薬Aの場合も $x = 0$ の薬Bの場合も男女全体を見ると半々で効果があり, 男女全体では効果に変わりがないことを意味している.

(5) 薬Aと薬Bのどちらを与えたかと男女のどちらであるかは独立である. そのことは効果の有無を意味する $y = 1, 0$ について和を取って得られる確率質量函数 $P(x, z)$ の表

	$z = 1$	$z = 0$
$x = 1$	$P(x = 1, z = 1) = 1/4$	$P(x = 1, z = 0) = 1/4$
$x = 0$	$P(x = 0, z = 1) = 1/4$	$P(x = 0, z = 0) = 1/4$

からわかる.

(6) 薬Aと薬Bのどちらを与えたかを無視すると, 男女のどちらであるかと薬の効果の有無は独立である. そのことは薬Aと薬Bのどちらを与えたかを意味する $x = 1, 0$ について和を取って得られる確率質量函数 $P(y, z)$ の表

	$z = 1$	$z = 0$
$y = 1$	$P(y = 1, z = 1) = 1/4$	$P(y = 1, z = 0) = 1/4$
$y = 0$	$P(y = 0, z = 1) = 1/4$	$P(y = 0, z = 0) = 1/4$

からわかる.

(7) しかし, 男女を区別すると全然違う結果が見えて来る. 薬Aは女性だけに効果があり, 薬Bは男性だけに効果がある. $z = 1$ の女性の場合に制限した確率質量函数の表

	$z = 1$	
	$y = 1$	$y = 0$
$x = 1$	$P(1, 1, 1) = 1/4$	$P(1, 0, 1) = 0$
$x = 0$	$P(0, 1, 1) = 0$	$P(0, 0, 1) = 1/4$

より, $x = 1$ の薬Aの場合には $y = 1$ の確率が正で効果があるが, $x = 0$ の薬Bの場合には $y = 1$ の確率が0で効果がないことがわかる. $z = 0$ の男性の場合に制限した確率質量函数の表

	$z = 0$	
	$y = 1$	$y = 0$
$x = 1$	$P(1, 1, 0) = 0$	$P(1, 0, 0) = 1/4$
$x = 0$	$P(0, 1, 0) = 1/4$	$P(0, 0, 0) = 0$

より, $x = 1$ の薬Aの場合には $y = 1$ の確率が0で効果がないが, $x = 0$ の薬Bの場合には $y = 1$ の確率が正で効果があることがわかる.

このように確率変数達が独立か否かは現実において重大な意味を持ち得る. ある重要な条件(上の場合には女性か男性か)を無視して, 「XとYは独立である」と結論すると大変なことになってしまう場合がある. XとYも, XとZも, YとZも独立であっても, XとYとZの全体は独立でないかもしれない.

2 無相関性

2.1 共分散と相関係数の定義および無相関の定義

確率函数 X, Y の期待値をそれぞれ $\mu_X = E[X]$, $\mu_Y = E[Y]$ と書くことにする.

確率変数 X, Y の **共分散** (covariance) $\sigma_{XY} = \mathrm{cov}(X, Y)$ を次のように定義する:

$$\sigma_{XY} = \mathrm{cov}(X, Y) = E[(X - \mu_X)(Y - \mu_Y)].$$

$X = Y$ のときこれは X の分散になる.

線形代数の言葉を使えば, 共分散 $\sigma_{XY} = \mathrm{cov}(X, Y)$ は内積に対応しており, 分散 $\sigma_X^2 = \mathrm{var}(X)$, $\sigma_Y^2 = \mathrm{var}(Y)$ はノルム(ベクトルの長さ)の二乗に対応しており, 標準偏差 $\sigma_X = \mathrm{std}(X)$, $\sigma_Y = \mathrm{std}(Y)$ はノルム(ベクトルの長さ)に対応している.

さらに確率変数 X, Y の **相関係数** (correlation coefficient) $\rho_{XY} = \mathrm{cov}(X, Y)$ はベクトルのあいだのなす角度 θ に対する $\cos\theta$ の対応物として次のように定義される:

$$\rho_{XY} = \mathrm{cor}(X, Y) = \frac{\mathrm{cov}(X, Y)}{\mathrm{std}(X)\,\mathrm{std}(Y)} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}.$$

相関係数という言葉を見たら $\cos\theta$ を想像し, 分散, 標準偏差, 共分散という言葉を見たら, ベクトルの長さの2乗, ベクトルの長さ, ベクトルの内積を想像すればよい. $X - \mu_X$ や $Y - \mu_Y$ がベクトルに対応している.

X, Y の共分散が $\mathrm{cov}(X, Y) = 0$ のとき(この条件は相関係数が $\mathrm{cor}(X, Y) = 0$ となることと同値, 直観的には「直交している」と考える), 確率変数達 X, Y は **無相関** であるという.

確率変数達 $X_1, \ldots, X_n$ が無相関であるとは, そのうちの互いに異なる任意の2つが無相関であることだと定める. $\mu_i = E[X_i]$, $\sigma_i^2 = \mathrm{var}(X_i)$ のとき, $X_1, \ldots, X_n$ が無相関であることは,

$$\mathrm{cov}(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)] = \sigma_i^2 \delta_{ij}$$

と書けることと同値である. ここで δ_{ij} は $i = j$ の場合にのみ 1 になり, それ以外の場合に 0 になるKroneckerのデルタである.

2.2 問題: 独立ならば無相関である (実質1行で解ける)

確率変数達 X, Y が独立ならば確率変数達 X, Y は無相関であることを示せ.

解答例: X, Y は独立なので $E[f(X)g(Y)] = E[f(X)]E[g(Y)]$ となる. ゆえに, $\mu_X = E[X]$, $\mu_Y = E[Y]$ とおくと,

$$E[(X - \mu_X)(Y - \mu_Y)] = E[X - \mu_X]E[Y - \mu_Y] = (E[X] - \mu_X)(E[Y] - \mu_Y) = 0 \cdot 0 = 0.$$

解答終

2.3 問題: 無相関でも独立とは限らない

確率変数 X, Y で無相関だが独立でないものを具体的に構成せよ.

解答例1: 確率質量函数 $P(x, y)$ を次の表の通りに定める:

	$y = 1$	$y = 2$	$y = 3$	
$x = 1$	$P(1,1) = 0$	$P(1,2) = 1/4$	$P(1,3) = 0$	$P(x = 1) = 1/4$
$x = 2$	$P(2,1) = 1/4$	$P(2,2) = 0$	$P(2,3) = 1/4$	$P(x = 2) = 2/4$
$x = 3$	$P(3,1) = 0$	$P(3,2) = 1/4$	$P(3,3) = 0$	$P(x = 3) = 1/4$
	$P(y = 1) = 1/4$	$P(y = 2) = 2/4$	$P(y = 3) = 1/4$	

確率変数 X, Y は同時確率質量函数 $P(x, y)$ を持つとする. このとき,

$$E[X] = E[Y] = 1 \cdot (1/4) + 2 \cdot (2/4) + 3 \cdot (1/4) = 2.$$

であり, 確率が 0 でない (x, y) については x または y が X と Y の期待値 2 になるので, 共分散は 0 になる:

$$\begin{aligned}\mathrm{cov}(X, Y) = E[(X - 2)(Y - 2)] &= \sum_{x,y} (x - 2)(y - 2)P(x, y) \\ &= (2 - 2)(1 - 2)(1/4) + (1 - 2)(2 - 2)(1/4) \\ &\quad + (3 - 2)(2 - 2)(1/4) + (2 - 2)(3 - 2)(1/4) = 0.\end{aligned}$$

これで X, Y は無相関であることがわかった.

しかし, $P(1, 1) = 0$ と $P(x = 1)P(y = 1) = (1/4)(1/4)$ は等しくないので X, Y は独立ではない.

解答終

解答例2: 確率質量函数 $P(x, y)$ を次の表の通りに定めても上と同様に, X, Y は無相関だが独立ではないことを示せる:

	$y = 1$	$y = 2$	$y = 3$	
$x = 1$	$P(1,1) = 1/8$	$P(1,2) = 1/8$	$P(1,3) = 1/8$	$P(x = 1) = 3/8$
$x = 2$	$P(2,1) = 1/8$	$P(2,2) = 0$	$P(2,3) = 1/8$	$P(x = 2) = 2/8$
$x = 3$	$P(3,1) = 1/8$	$P(3,2) = 1/8$	$P(3,3) = 1/8$	$P(x = 3) = 3/8$
	$P(y = 1) = 3/8$	$P(y = 2) = 2/8$	$P(y = 3) = 3/8$	

解答終

解答例3: $\mathbb{R}^2$ における単位円盤上の一様分布の確率密度函数を次のように定める:

$$p(x,y) = \begin{cases} 1/\pi & (x^2+y^2 \le 1) \\ 0 & (x^2+y^2 > 1) \end{cases}$$

これを同時確率密度函数として持つ確率変数 X, Y を考える:

$$E[f(X,Y)] = \iint_{\mathbb{R}^2} f(x,y)p(x,y)\,dx\,dy.$$

単位円盤の対称性から $E[X] = E[Y] = 0$ となることがわかる. (具体的に積分を計算しても易しい.) それらの共分散は

$$\mathrm{cov}(X,Y) = E[XY] = \frac{1}{\pi}\iint_{x^2+y^2\le 1} xy\;dx\,dy$$

と書けるが, $xy \ge 0$ の部分の積分と $xy \le 0$ の部分の積分が円盤の対称性より互いに打ち消しあって $\mathrm{cov}(X,Y) = 0$ となることがわかる. X 単独の密度函数は

$$p(x) = \frac{1}{\pi}\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} dy = \frac{2}{\pi}\sqrt{1-x^2}$$

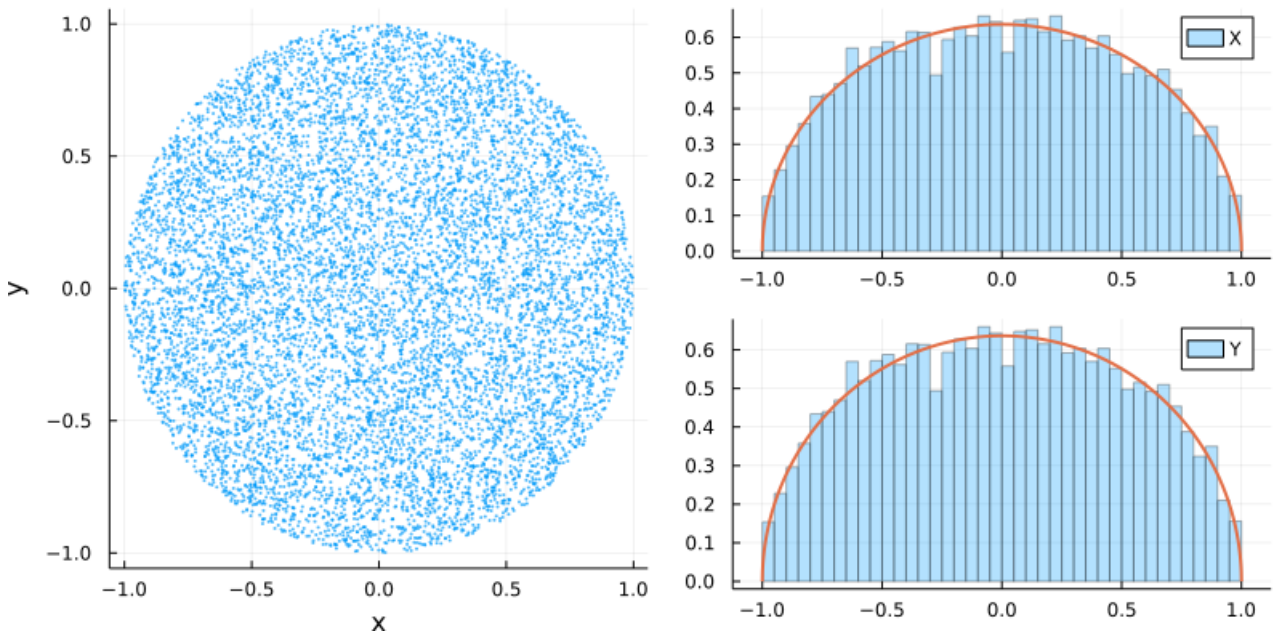
となり, 同様にして $p(y) = (2/\pi)\sqrt{1-y^2}$ となるが, $-1 < x, y < 1$ かつ $x^2+y^2 > 1$ のとき, $p(x,y) = 0$ となるが, $p(x)p(y) \ne 0$ となるので, それらは等しくない. ゆえに X, Y は独立ではない.

解答終

In [28]:
```
n = 10^4
XY = [(r = √rand(); t = 2π*rand(); (r*cos(t), r*sin(t))) for _ in 1:n]
X, Y = first.(XY), last.(XY)
@show cov(X, Y)
P1 = scatter(X, Y; msc=:auto, ms=2, alpha=0.7, label="", xlabel="x", ylabel="y")
P2 = histogram(X; norm=true, alpha=0.3, bin=41, label="X")
plot!(x -> 2/π*√(1 - x^2), -1, 1; label="", lw=2)
P3 = histogram(X; norm=true, alpha=0.3, bin=41, label="Y")
plot!(y -> 2/π*√(1 - y^2), -1, 1; label="", lw=2)
plot(P1, P2, P3; size=(800, 400), layout=@layout [a [b; c]])
```

```
cov(X, Y) = -0.0005083072927225841
```

Out[28]:



単位円盤上の一様分布は「無相関だが独立ではない場合」の例になっている.

2.4 問題: 無相関な確率変数達の和の分散

$X_1, \ldots, X_n$ は独立でも無相関とも限らない確率変数達であるとする. このとき, 期待値を取る操作の線形性より,

$$E[X_1 + \cdots + X_n] = E[X_1] + \cdots + E[X_n]$$

となる. 確率変数達の和の期待値は, 独立性や無相関性が成立していなくても, それぞれの期待値の和になる.

無相関性を仮定すると, 分散についても簡明な結果を得ることができる.

$X_1, \dots, X_n$ が **無相関な確率変数達** ならば(特に独立な確率変数ならば), 次が成立することを示せ:

$$\mathrm{var}(X_1 + \cdots + X_n) = \mathrm{var}(X_1) + \cdots + \mathrm{var}(X_n).$$

この結果は今後空気のごとく使われる.

ヒント: 互いに直交するベクトル達 $v_1, \dots, v_n$ について, 内積を $(\ ,\ )$ と書くとき, $(v_i, v_j) = \delta_{ij}\|v_i\|^2$ が成立することを使えば, $\|v_1 + \cdots + v_n\|^2 = \|v_1\|^2 + \cdots + \|v_n\|^2$ を示せることと本質的に同じ. この結果はPythagorasの定理(平面の場合は三平方の定理)そのものである.

解答例: $\mu_i = E[X_i]$ とおくと,

$$E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} \mu_i$$

が成立しており, さらに, $\sigma_i^2 = \mathrm{var}(X_i) = E[(X_i - \mu_i)^2]$ とおくと, $X_1, \dots, X_n$ が無相関であることより,

$$E[(X_i - \mu_i)(X_j - \mu_j)] = \mathrm{cov}(X_i, X_j) = \sigma_i^2 \delta_{ij}$$

が成立しているので(δ_{ij} は $i = j$ の場合にのみ 1 でそれ以外のとき 0), 一般に

$$\left(\sum_{i=1}^{n} a_i\right)^2 = \sum_{i=1}^{n} a_i \sum_{i=j}^{n} a_j = \sum_{i,j=1}^{n} a_i a_j$$

と計算できることを使うと,

$$\begin{aligned}
\mathrm{var}\left(\sum_{i=1}^{n} X_i\right) &= E\left[\left(\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} \mu_i\right)^2\right] = E\left[\left(\sum_{i=1}^{n} (X_i - \mu_i)\right)^2\right] \\
&= E\left[\sum_{i,j=1}^{n} (X_i - \mu_i)(X_j - \mu_j)\right] = \sum_{i,j=1}^{n} E\left[(X_i - \mu_i)(X_j - \mu_j)\right] \\
&= \sum_{i,j=1}^{n} \sigma_i^2 \delta_{ij} = \sum_{i=1}^{n} \sigma_i^2 = \sum_{i=1}^{n} \mathrm{var}(X_i).
\end{aligned}$$

解答終

2.5 問題: 二項分布と負の二項分布の平均と分散のBernoulli分布と幾何分布の場合への帰着

Bernoulli分布 $\mathrm{Bernoulli}(p)$ の平均と分散がそれぞれ p, $p(1-p)$ であることと, 幾何分布 $\mathrm{Geometric}(p)$ の平均と分散がそれぞれ $(1-p)/p$, $(1-p)/p^2$ であることを認めて, 二項分布 $\mathrm{Binomial}(n, p)$ と負の二項分布 $\mathrm{NegativeBinomial}(k, p)$ の平均と分散を平易な計算で求めてみよ. 以下を示せ:

(1) $K \sim \mathrm{Binomial}(n, p)$ ならば $E[K] = np$, $\mathrm{var}(K) = np(1-p)$.

(2) $M \sim \mathrm{NegativeBinomial}(k, p)$ ならば $E[M] = k(1-p)/p$, $\mathrm{var}(M) = k(1-p)/p^2$.

解答例:

二項分布は試行回数 n の成功確率 p のBernoulli試行で生成された 1 と 0 からなる長さ n の数列中に含まれる 1 の個数の分布であった. Bernoulli試行の確率質量函数は

$$P(x_1, \dots, x_n) = P(x_1) \cdots P(x_n), \quad P(x_i) = p^{x_i}(1-p)^{1-x_i} \quad (x_i = 1, 0)$$

とBernoulli分布の確率質量函数 $P(x_i)$ の積で書けるのであった. この事実はBernoulli分布に従う確率変数 $X_1, \dots, X_n$ が独立であることを意味する. そして, Bernoulli試行で生成された 1 と 0 からなる長さ n の数列中に含まれる 1 の個数を意味する確率変数は $K = \sum_{i=1}^{n} X_i$ と書ける. このことから,

$$\begin{aligned}
E[K] &= \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} p = np, \\
\mathrm{var}(K) &= \sum_{i=1}^{n} \mathrm{var}(X_i) = \sum_{i=1}^{n} p(1-p) = np(1-p).
\end{aligned}$$

となることがわかる.

幾何分布は成功確率 p のBernoulli試行を 1 が1つ出るまで続けたときに出た 0 の個数の分布であった. $M_1, \dots, M_k$ はそれぞれが成功確率 p の幾何分布に従う独立な確率変数であるとする. このとき, $M = \sum_{i=1}^{k} M_i$ はBernoulli試行を 1 が k 回出るまで続けたときに 0 が出た個数に等しい. M_i は $i-1$ 番目の 1 から i 番目の 1 が出るまでに出た 0 の個数を意味する確率変数だと解釈される. このことは, M が負の二項分布 $\mathrm{NegativeBinomial}(k, p)$ に従う確率変数になることを意味する. このことから,

$$E[M] = \sum_{i=1}^{k} E[M_i] = \sum_{i=1}^{k} \frac{1-p}{p} = \frac{k(1-p)}{p},$$
$$\mathrm{var}(M) = \sum_{i=1}^{k} \mathrm{var}(M_i) = \sum_{i=1}^{k} \frac{1-p}{p^2} = \frac{k(1-p)}{p^2}.$$

となることがわかる.

解答終

注意: 計算が大幅に簡単になった!

2.6 問題: 番号が異なる確率変数達が無相関なときの確率変数の和の共分散

確率変数達 $X_1, Y_1, \dots, X_n, Y_n$ について次が成立していると仮定する:

$$\mathrm{cov}(X_i, Y_j) = \delta_{ij} \mathrm{cov}(X_i, Y_i).$$

このとき, 次が成立することを示せ:

$$\mathrm{cov}(X_1 + \cdots + X_n, Y_1 + \cdots + Y_n) = \mathrm{cov}(X_1, Y_1) + \cdots + \mathrm{cov}(X_n, Y_n).$$

ヒント: ベクトル達 $u_1, v_2, \dots, u_n, v_n$ の中の2つの異なる添え字を持つ u_i と v_j が互いに直交するならば, 内積 $(\ ,\ )$ について

$$(u_1 + \cdots + u_n, v_1 + \cdots + v_n) = (u_1, v_1) + \cdots + (u_n, v_n)$$

が成立することと本質的に同じことである.

解答例:

記号の簡単のため $A_i = X_i - E[X_i]$, $B_i = Y_i - E[Y_i]$ とおく. このとき, $E[A_i] = E[B_i] = 0$ より,

$$\mathrm{var}(A_i) = E[A_i^2], \quad \mathrm{var}(B_i) = E[B_i^2], \quad \mathrm{cov}(A_i, B_j) = E[A_i B_j].$$

A_i, B_i 達について上の問題を解けばよい. 問題の仮定より, A_i, B_i 達について次が成立している:

$$E[A_i B_j] = \delta_{ij} E[A_i B_i].$$

ゆえに,

$$E\left[\left(\sum_{i=1}^{n} A_i\right)\left(\sum_{j=1}^{n} B_j\right)\right] = \sum_{i,j=1}^{n} E[A_i B_j] = \sum_{i,j=1}^{n} \delta_{ij} E[A_i B_i] = \sum_{i=1}^{n} E[A_i B_i].$$

これは

$$\mathrm{cov}(A_1 + \cdots + A_n, B_1 + \cdots + B_n) = \mathrm{cov}(A_1, B_1) + \cdots + \mathrm{cov}(A_n, B_n).$$

の成立を意味する.

解答終

3 モーメントとその母函数と特性函数とキュムラント母函数

3.1 モーメントとその母函数と特性函数とキュムラント母函数の定義

確率変数 X と $m = 0, 1, 2, \dots$ について

$$\mu_m(X) = E[X^m]$$

を X の m 次の **モーメント**(moment, 積率) と呼び,

$$M_X(t) = E[e^{tX}] = E\left[\sum_{m=0}^{\infty} X^m \frac{t^m}{m!}\right] = \sum_{m=0}^{\infty} E[X^m] \frac{t^m}{m!} = \sum_{m=0}^{\infty} \mu_m(X) \frac{t^m}{m!}$$

を **モーメント母函数** (moment generating function, mgf)と呼ぶ.

X が従う確率分布の名前が Dist のとき, これらを **分布** Dist **のモーメントとモーメント母函数** と呼ぶ. 以下も同様である.

モーメント母函数の定義で t を $it = \sqrt{-1}\,t$ で置き換えたもの

$$\varphi_X(t) = E[e^{itX}] = E\left[\sum_{m=0}^{\infty} i^m X^m \frac{t^m}{m!}\right] = \sum_{m=0}^{\infty} i^m E[X^m]\frac{t^m}{m!} = \sum_{m=0}^{\infty} i^m \mu_m(X)\frac{t^m}{m!}$$

を **特性函数** (characteristic function)と呼ぶ. 特性函数を扱う場合には $i = \sqrt{-1}$ としたいので, i を番号の意味で使わないように気を付ける必要がある.

モーメント母函数だけではなく, 特性函数もモーメント達の母函数になっている.

モーメント母函数の対数

$$K_X(t) = \log M(t) = \log E[e^{tX}] = \sum_{m=1}^{\infty} \kappa_m(X)\frac{t^m}{m!} = \sum_{m=1}^{\infty} \kappa_m \frac{t^m}{m!}$$

を **キュムラント母函数** (cumulant generating function, cgf) と呼び, その展開係数 $\kappa_m = \kappa_m(X)$ を X の m 次のキュムラントと呼ぶ.

注意: 取り得る値が実数になる確率変数 X について $|e^{itX}| = 1$ となるので, $E[e^{itX}]$ は常に絶対収束しており, 特性函数は常にうまく定義されている. それに対して e^{tX} の値は巨大になる可能性があり, $E[e^{tX}]$ が収束しない場合が出て来る. モーメント母函数やキュムラント母函数の取り扱いではこの点に注意する必要がある.

注意: モーメント母函数とキュムラント母函数はそれぞれ物理での統計力学における **分配函数** と **自由エネルギー** (もしくは **Massieu函数**)の確率論的な類似物になっている. ただし, 逆温度 β について $t = -\beta$ とおく必要がある. 逆に言えば, モーメント母函数とキュムラントの表示における $\beta = -t$ の逆数は絶対温度の確率論的類似物になっていることになる.

3.2 特性函数による期待値の表示

X が確率密度函数 $p(x)$ を持つとき, 函数 $f(x)$ のFourier変換を

$$\hat{f}(t) = \int_{-\infty}^{\infty} f(x)e^{-itx}\,dx$$

と書くと, $f(t)$ がそう悪くない函数ならば逆Fourier変換によってもとの函数に戻せる:

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(t)e^{itx}\,dt.$$

Fourier解析の基礎については次のリンク先を参照せよ(逆Fourier変換に関する結果はこのノート内では認めて使ってよい):

- 12 Fourier解析 (https://nbviewer.org/github/genkuroki/Calculus/blob/master/12%20Fourier%20analysis.ipynb)

ゆえに, x に確率変数 X を代入して両辺の期待値を取り, 期待値を取る操作と積分を交換すると,

$$E[f(X)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(t)E[e^{itX}]\,dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(t)\varphi_X(t)\,dt.$$

ここで $\varphi_X(t) = E[e^{itX}]$ は X の特性函数である.

確率変数 X が従う分布は様々な函数 $f(x)$ に関する期待値 $E[f(X)]$ から決まるので, $E[f(X)]$ が X の特性函数 $\varphi_X(t)$ を用いて表せたということは, X **の特性函数** $\varphi_X(t)$ **から** X **が従う分布が唯一つに決まる** ことを意味している.

さらに, 分布Aの特性函数が分布Bの特性函数で近似されていれば, 分布Aにおける期待値が分布Bにおける期待値で近似されることもわかる. これは **分布の近似を特性函数の近似で確認できる** ことを意味する.

モーメント母函数 $M_X(t) = E[e^{tX}]$ が t の函数として, 定義域を自然に複素数まで拡張できているとき(正確には解析接続できていれば), $\varphi_X(t) = M_X(it)$ が成立する. キュムラント母函数はモーメント母函数の対数である. これらより, **モーメント母函数やキュムラント母函数からも分布が唯一つに決まる** ことがわかる.

3.3 問題: キュムラントのロケーションスケール変換

確率変数 X と $a, b \in \mathbb{R}$, $a \neq 0$ について, $aX + b$ のモーメント母函数とキュムラント母函数とキュムラントが X のそれらで次のように表されることを示せ:

$$M_{aX+b}(t) = e^{bt} M_X(at), \quad K_{aX+b}(t) = K_X(at) + bt,$$
$$\kappa_1(aX+b) = a\kappa_1(X) + b, \quad \kappa_m(aX+b) = a^m \kappa_m(X) \quad (m = 2, 3, 4, \ldots)$$

注意: キュムラントの変換公式は非常に単純な形になる. $\kappa_1(aX+b) = a\kappa_1(X) + b$ は $\kappa_1(X) = E[X]$ だったので当然である. 2次以上のキュムラントは a^m 倍されるだけになる. モーメント母函数の対数を取ってキュムラント母函数を定義し, その展開によってキュムラントを定義することにはこのような利点がある. キュムラント母函数を定義することには, 物理の統計力学で分配函数の対数を取って自由エネルギーを定義することと同様の利点がある.

注意: この結果は空気のごとく使われる.

解答例: $aX+b$ のモーメント母函数を X のモーメント母函数であらわそう:

$$M_{aX+b}(t) = E[e^{t(aX+b)}] = e^{bt} E[e^{atX}] = e^{bt} M_X(at).$$

ゆえに, $aX+b$ のキュムラントは次の形になる:

$$K_{aX+b}(t) = \log M_{aX+b}(t) = \log(e^{bt} M_X(at)) = K_X(at) + bt.$$

$X \mapsto aX+b$ によってキュムラント母函数は $K_X(t) \mapsto K_X(at) + bt$ に似た形式で変換される.

X のキュムラント $\kappa_m(X)$ は次のようにキュムラント母函数を展開することによって定義されるのであった:

$$K_X(t) = \sum_{m=1}^{\infty} \kappa_m(X) \frac{t^m}{m!}.$$

$K_{aX+b}(t)$ の展開結果は

$$K_{aX+b}(t) = K_X(at) + bt = \sum_{m=0}^{\infty} \kappa_m(X) \frac{(at)^m}{m!} + bt = (a\kappa_1(X) + b)t + \sum_{m=2}^{\infty} a^m \kappa_m(X) \frac{t^m}{m!}$$

になるので,

$$\kappa_1(aX+b) = a\kappa_1(X) + b, \quad \kappa_m(aX+b) = a^m \kappa_m(X) \quad (m = 2, 3, 4, \ldots)$$

となることがわかる.

解答終

3.4 問題: 標準正規分布のモーメント母函数と特性函数とキュムラント母函数

標準正規分布に従う確率変数 Z のモーメント母函数と特性函数とキュムラント母函数が次のようになることを示せ:

$$M_Z(t) = e^{t^2/2}, \quad \varphi_Z(t) = M_Z(it) = e^{-t^2/2}, \quad K_Z(t) = \log M_Z(t) = \frac{t^2}{2}.$$

この結果は中心極限定理の証明で使われる.

解答例: $Z \sim \mathrm{Normal}(0,1)$ と仮定する. このとき,

$$tz - \frac{z^2}{2} = -\frac{z^2 - 2tz}{2} = -\frac{(z-t)^2 - t^2}{2} = -\frac{(z-t)^2}{2} + \frac{t^2}{2}$$

なので,

$$M_Z(t) = E[e^{tZ}] = \int_{-\infty}^{\infty} e^{tz} \frac{e^{-z^2/2}}{\sqrt{2\pi}}\, dz = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-(z-t)^2/2 + t^2/2}\, dz = \frac{e^{t^2/2}}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-w^2/2}\, dw = e^{t^2/2}.$$

4つめの等号で $z = w + t$ とおいた. ゆえに,

$$\varphi_Z(t) = E[e^{itZ}] = M_Z(it) = e^{-t^2/2}, \quad K_Z(t) = \log M_Z(t) = \frac{t^2}{2}.$$

解答終

注意: 標準正規分布のキュムラント母函数は $t^2/2$ というたったの一項だけになってしまう. 標準でない正規分布のキュムラント母函数は t について1次と2次の項だけになる. 1次の項の係数は分布の期待値で, $t^2/2$ の係数は分散になる. 実際, 平均 μ, 分散 σ^2 を持つ確率変数 X について,

$$
\begin{aligned}
K_X(t) &= \log E[e^{tX}] = \log E[e^{t(X-\mu)+t\mu}] \\
&= t\mu + \log\left(1 + E[X-\mu]t + E[(X-\mu)^2]\frac{t^2}{2} + O(t^3)\right) \\
&= t\mu + \log\left(1 + \sigma^2\frac{t^2}{2} + O(t^3)\right) = t\mu + \sigma^2\frac{t^2}{2} + O(t^3).
\end{aligned}
$$

そして, $\sigma Z + \mu \sim \mathrm{Normal}(\mu, \sigma)$ であり,

$$
M_{\sigma Z+\mu}(t) = E[e^{t(\sigma Z+\mu)}] = e^{\mu t}\, E[e^{t\sigma Z}] = e^{\mu t}\, M_Z(\sigma t) = e^{\mu t + \sigma^2 t^2/2}.
$$

ゆえに

$$
K_{\sigma Z+\mu}(t) = \log M_{\sigma Z+\mu}(t) = \mu t + \sigma^2\frac{t^2}{2}.
$$

他の分布のキュムラント母函数を計算したときに出て来る t について3次以上の項はその分布が正規分布からどれだけ離れているかを表している.

3.5 確率変数の標準化と標準化キュムラントと歪度と尖度

確率変数 X は確率変数であるとし, $\mu = E[X]$, $\sigma = \sqrt{E[(X-\mu)^2]}$ とおく. このとき,

$$
Z = \frac{X-\mu}{\sigma}
$$

を確率変数の **標準化** (standardization)と呼ぶ. Z の期待値と分散はそれぞれ 0 と 1 になる.

X の標準化のモーメントやキュムラントをそれぞれ **標準化モーメント**, **標準化キュムラント** と呼び, それぞれを $\bar{\mu}_m(X)$, $\bar{\kappa}_m(X)$ と表す. 詳しくは以下の通り:

$$
\begin{aligned}
&\bar{\mu}_m(X) = \mu_m(Z) = E\left[\left(\frac{X-\mu}{\sigma}\right)^m\right], \\
&M_Z(t) = E\left[\exp\left(t\frac{X-\mu}{\sigma}\right)\right] = \sum_{m=0}^{\infty}\bar{\mu}_m(X)\frac{t^m}{m!} = 1 + \frac{t^2}{2} + \bar{\mu}_3(X)\frac{t^3}{3!} + \bar{\mu}_4(X)\frac{t^4}{4!} + \cdots, \\
&K_Z(t) = \log M_Z(t) = \sum_{m=1}^{\infty}\bar{\kappa}_m(X)\frac{t^m}{m!} = \frac{t^2}{2} + \bar{\kappa}_3(X)\frac{t^3}{3!} + \bar{\kappa}_4(X)\frac{t^4}{4!} + \cdots.
\end{aligned}
$$

$\bar{\kappa}_3(X)$ と $\bar{\kappa}_4(X)$ は次のように表される:

$$
\bar{\kappa}_3(X) = \bar{\mu}_3(X) = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right], \quad \bar{\kappa}_4(X) = \bar{\mu}_4(X) - 3 = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] - 3.
$$

このことは, $\log(1+a) = a - a^2/2 + O(a^3)$ を使って以下のようにして確認される:

$$
\begin{aligned}
\log\left(1 + \frac{t^2}{2} + \bar{\mu}_3(X)\frac{t^3}{3!} + \bar{\mu}_4(X)\frac{t^4}{4!} + O(t^5)\right) &= \frac{t^2}{2} + \bar{\mu}_3(X)\frac{t^3}{3!} + \bar{\mu}_4(X)\frac{t^4}{4!} - \frac{1}{2}\left(\frac{t^2}{2}\right)^2 + O(t^5) \\
&= \frac{t^2}{2} + \bar{\mu}_3(X)\frac{t^3}{3!} + (\bar{\mu}_4(X) - 3)\frac{t^4}{4!} + O(t^5).
\end{aligned}
$$

$\bar{\kappa}_3(X)$ を X もしくは X が従う分布の **歪度** (わいど, skewness) と呼び, $\bar{\kappa}_4(X)$ を **尖度** (せんど, kurtosis)と呼び, 次のようにも書くことにする:

$$
\mathrm{skewness}(X) = \bar{\kappa}_3(X) = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right], \quad \mathrm{kurtosis}(X) = \bar{\kappa}_4(X) = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] - 3.
$$

歪度は左右の非対称性の尺度であり, 尖度は分布の尖り具合が正規分布とどれだけ違うかの尺度になっている.

X が正規分布に従う確率変数の場合にはその標準化 $Z = (X-\mu)/\sigma$ は標準正規分布に従う確率変数になるので, その標準化キュムラント達は $\bar{\kappa}_2(X) = 1$, $\bar{\kappa}_m(X) = 0$ $(m \neq 0)$ となる. 2次の標準化キュムラントは常に 1 になるが, 3 次以上の標準化キュムラントは X が正規分布でなければ 0 でなくなる.

このことから, 3 次以上の標準化キュムラントは分布が正規分布からどれだけ離れているかを表していると考えられる. それらのうち最初の2つが上で定義した歪度 $\bar{\kappa}_3(X)$ と尖度 $\bar{\kappa}_4(X)$ になっている.

分布の歪度 $\bar{\kappa}_3(X)$ と尖度 $\bar{\kappa}_4(X)$ は分布がどれだけ正規分布から離れているかを表す最も基本的な量である.

注意: $\bar{\kappa}_4(X) = \bar{\mu}_4(X) - 3$ ではなく, 3 を引く前の $\bar{\mu}_4(X)$ を尖度と定義する流儀もあるが, このノートでは正規分布の扱いでキュムラントが非常に便利なことを重視したいので, 3 を引いた側の $\bar{\kappa}_4$ を尖度の定義として採用する. 3 を引いた方の $\bar{\kappa}_4(X) = \bar{\mu}_4(X) - 3$ は **過剰尖度** (かじょうせんど, **excess kurtosis**)と呼ばれることも多い. 正規分布の尖度を($\bar{\kappa}_4$ の方の 0 ではなく) $\bar{\mu}_4$ の方の 3 とするときに, そこからどれだけ分布の尖り具合が増したかを $\bar{\kappa}(X)$ が表していることを「過剰」と表現している.

In [29]:
```
@vars t μ3 μ4 μ5 κ3 κ4 κ5
Mt = 1 + t^2/2 + μ3*t^3/6 + μ4*t^4/24 + μ5*t^5
expr = series(log(Mt), t)
```

Out[29]:
$$\frac{t^2}{2} + \frac{t^3\mu_3}{6} + t^4\left(\frac{\mu_4}{24} - \frac{1}{8}\right) + t^5\left(-\frac{\mu_3}{12} + \mu_5\right) + O\left(t^6\right)$$

3.6 問題: 独立な確率変数達の和のモーメント母函数と特性函数とキュムラント母函数

独立な確率変数達 $X_1, \ldots, X_n$ の和のモーメント母函数と特性函数とキュムラント母函数が次のように表されることを示せ:

$$\begin{aligned}
M_{X_1+\cdots+X_n}(t) &= M_{X_1}(t)\cdots M_{X_n}(t),\\
\varphi_{X_1+\cdots+X_n}(t) &= \varphi_{X_1}(t)\cdots \varphi_{X_n}(t),\\
K_{X_1+\cdots+X_n}(t) &= K_{X_1}(t) + \cdots + K_{X_n}(t).
\end{aligned}$$

注意: この結果は空気のごとく使われる.

解答例: 独立な確率変数達 $X_1, \ldots, X_n$ について

$$E[f_1(X_1)\cdots f_n(X_n)] = E[f_1(X_1)]\cdots E[f_n(X_n)]$$

が成立することより,

$$\begin{aligned}
M_{X_1+\cdots+X_n}(t) &= E[e^{t(X_1+\cdots+X_n)}] = E[e^{tX_1}\cdots e^{tX_n}] = M_{X_1}(t)\cdots M_{X_n}(t),\\
\varphi_{X_1+\cdots+X_n}(t) &= E[e^{it(X_1+\cdots+X_n)}] = E[e^{itX_1}\cdots e^{itX_n}] = \varphi_{X_1}(t)\cdots \varphi_{X_n}(t).
\end{aligned}$$

ゆえに

$$\begin{aligned}
K_{X_1+\cdots+X_n}(t) &= \log M_{X_1+\cdots+X_n}(t) = \log(M_{X_1}(t)\cdots M_{X_n}(t))\\
&= \log M_{X_1}(t) + \cdots + \log M_{X_n}(t) = K_{X_1}(t) + \cdots + K_{X_n}(t).
\end{aligned}$$

解答終

In [ ]:
```
```