r×cの分割表におけるPearsonのχ²統計量

黒木玄

2019-10-29〜2019-11-04, 2020-05-27

- ipynb版 (https://nbviewer.jupyter.org/github/genkuroki/Statistics/blob/master/Pearson%27s+%CF%87%C2%B2-statistics+of+contingency+tables.ipynb)
- pdf版 (https://genkuroki.github.io/documents/Statistics/Pearson%27s+%CF%87%C2%B2-statistics+of+contingency+tables.pdf)

このノートを書いたモチベーションについては以下のリンク先を参照:

- https://twitter.com/genkuroki/status/1189924642292551680 (https://twitter.com/genkuroki/status/1189924642292551680)
- https://twitter.com/genkuroki/status/1190254626072748032 (https://twitter.com/genkuroki/status/1190254626072748032)

目次

1 分割表の確率分布

非負の整数からなる $r\times c$ 行列に値を持つ確率変数を**分割表 (contingency table)** と呼ぶ. 以下では, $r\times c$ の分割表を $r\times c$ の行列 $A=[a_{ij}]$ で表す. 以下において i は $1,\dots,r$ を走り, j は $1,\dots,c$ を走るものとする.

このノートでは分割表の確率分布として以下の4種類を考える. そして, サンプルを生成する分布としては, 独立性の条件を満たすものを考える.

1.1 rc 個のPoisson分布の直積

非負の整数成分の $r \times c$ の分割表 $A = [a_{ij}]$ に何も制限を課さない場合.

$\Lambda = [\lambda_{ij}]$ は正の実数を成分とする $r \times c$ 行列であるとし,

$$\lambda = \sum_{i,j} \lambda_{ij}, \quad p_{ij} = \frac{\lambda_{ij}}{\lambda}, \quad P = [p_{ij}]$$

とおく. このとき分割表 $A = [a_{ij}]$ が生じる確率 $p(A|\Lambda)$ を

$$p(A|\Lambda) = \prod_{i,j} \frac{e^{-\lambda_{ij}} \lambda_{ij}^{a_{ij}}}{a_{ij}!}$$

と定めることができる. このようにして定まる分割表の確率分布を rc 個のPoisson分布の直積と呼ぶ.

この rc 個のPoisson分布の直積において, 各 a_{ij} の期待値は λ_{ij} になり, a_{ij} 達の総和の期待値は λ になる.

パラメーター $\Lambda = [\lambda_{ij}]$ もしくは $P = [p_{ij}]$ が独立性の条件を満たしているとは, p_{ij} 達が,

$$p_{ij} = p_i q_j, \quad p_i, q_j \geqq 0, \quad \sum_i p_i = \sum_j q_j = 1$$

と表わされることだと定める. この条件は

$$\mu_i = \lambda p_i, \quad \nu_j = \lambda q_j$$

と定めると,

$$\lambda_{ij} = \frac{\mu_i \nu_j}{\lambda}, \quad \mu_i, \nu_j \geqq 0, \quad \sum_i \mu_i = \sum_j \nu_j = \lambda$$

と書き直される. このとき,

$$p(A|\Lambda) = \prod_{i,j} \frac{e^{-\mu_i \nu_j/\lambda} (\mu_i \nu_j/\lambda)^{a_{ij}}}{a_{ij}!}.$$

rc 個のPoisson分布の直積におけるパラメーター全体の空間の次元は rc であり, その中で独立性を満たすパラメーター達のなす部分空間の次元は λ の分の 1 と μ_i 達の分の $r-1$ と ν_j 達の分の $c-1$ の総和である $r+c-1$ 次元になり, パラメーター全体の空間との次元の差は $rc - r - c + 1 = (r-1)(c-1)$ になる. この $(r-1)(c-1)$ がχ²検定におけるχ²分布の自由度になる.

1.2 rc 項分布 (多項分布)

分割表 $A = [a_{ij}]$ に総和 $\sum_{i,j} a_{ij} = n$ が一定であるという制限を課す場合.

p_{ij} は非負の実数であるとし, それらの総和は 1 になると仮定し, $P = [p_{ij}]$ とおく.

非負の整数成分の $r \times c$ 行列を $r \times c$ の分割表と呼ぶのであった. 成分の総和が n に固定された $r \times c$ の分割表 $A = [a_{ij}]$, ($\sum_{i,j} a_{ij} = n$) が生じる確率を

$$p(A|n, P) = n! \prod_{i,j} \frac{p_{ij}^{a_{ij}}}{a_{ij}!}$$

と定めることによって, 成分の総和が n に固定された分割表全体に確率分布を定めることができる. これを rc 項分布と呼ぶことにする.

この rc 項分布における a_{ij} の期待値 λ_{ij} は

$$\lambda_{ij} = np_{ij}$$

になる. これを用いると, rc 項分布における確率は

$$p(A|n, P) = \frac{n!}{n^n} \prod_{i,j} \frac{\lambda_{ij}^{a_{ij}}}{a_{ij}!}$$

と書き直される.

この場合のパラメーター $P = [p_{ij}]$ に関する独立性の条件は

$$p_{ij} = p_i q_j, \quad p_i, q_j \geqq 0, \quad \sum_i p_i = \sum_j q_j = 1$$

もしくは

$$\lambda_{ij} = \frac{\mu_i \nu_j}{\lambda}, \quad \mu_i, \nu_j \geqq 0, \quad \sum_i \mu_i = \sum_j \nu_j = n$$

と書ける.

rc 項分布におけるパラメーター全体の空間の次元は $\sum_{i,j} p_{ij} = 1$ という制限によって rc より1小さい $rc - 1$ になり, 独立性を満たすパラメーター達のなす部分空間の次元は p_i 達の分の $r - 1$ と q_i 達の分の $c - 1$ の和の $r + c - 2$ になり, 全体の次元との差は $(r - 1)(c - 1)$ になる. この $(r - 1)(c - 1)$ がχ²検定におけるχ²分布の自由度になる.

1.3 r 個の c 項分布の直積

分割表 $A = [a_{ij}]$ に行の和 $\sum_j a_{ij} = \mu_j$ がすべて一定であるという制限を課す場合.

n, μ_i は非負の整数であるとし, $\sum_i mu_i = n$ と仮定し,

$$\mu = (\mu_1, \dots, \mu_r)$$

とおく. q_{ij} は非負の実数であるとし, $\sum_j q_{ij} = 1$ であると仮定し,

$$Q = [q_{ij}]$$

とおく.

各行の総和が μ_i になるという条件

$$\sum_j a_{ij} = \mu_i$$

という条件を満たす分割表 $A = [a_{ij}]$ が生じる確率を

$$p(A|\mu, Q) = \prod_i \left(\mu_i! \prod_j \frac{q_{ij}^{a_{ij}}}{a_{ij}!} \right)$$

と定めることによって, 各行の総和が μ_i になるという制限付きの分割表全体に確率分布を定義できる. これを r 個の c 項分布の直積と呼ぶことにする.

この rc 項分布にいて, 各 a_{ij} の期待値は λ_{ij} は

$$\lambda_{ij} = \mu_i q_{ij}$$

になる. これを用いると, r 個の c 項分布の直積における確率は

$$p(A|\mu, Q) = \prod_i \frac{\mu_i!}{\mu_i^{\mu_i}} \cdot \prod_{i,j} \frac{\lambda_{ij}^{a_{ij}}}{a_{ij}!}$$

と書き直される.

この場合の独立性の条件は

$$\nu_j = \sum_i \lambda_{ij}, \quad q_j = \frac{\nu_j}{n}$$

とおくと

$$q_{1j} = \cdots = q_{rj} = q_j$$

もしくは

$$\frac{\lambda_{1j}}{\mu_1} = \cdots = \frac{\lambda_{rj}}{\mu_r} = \frac{\nu_j}{n}$$

と書ける.

r 個の c 項分布におけるパラメーター全体の空間の次元は $r(c - 1)$ になり, 独立性を満たすパラメーター達のなす部分空間の次元は q_i 達の分の $c - 1$ になり, 全体の次元との差は $(r - 1)(c - 1)$ になる. この $(r - 1)(c - 1)$ がχ²検定におけるχ²分布の自由度になる.

1.4 周辺度数がすべて固定されている分割表の確率分布

分割表 $A = [a_{ij}]$ に行の和 $\sum_j a_{ij} = \mu_j$ と列の和 $\sum_i a_{ij} = \nu_j$ がすべて一定であるという制限を課す場合.

n, μ_i, ν_j は正の整数で

$$\sum_i \mu_i = \sum_j \nu_j = n$$

を満たしていると仮定し,

$$\mu = (\mu_1, \dots, \mu_r), \quad \nu = (\nu_1, \dots, \nu_c)$$

とおく. λ_{ij} は正の実数であるとし,

$$\sum_j \lambda_{ij} = \mu_i, \quad \sum_i \lambda_{ij} = \nu_i$$

を満たしていると仮定し,

$$\Lambda = [\lambda_{ij}]$$

とおく.

すべての行とすべての列の総和が

$$\sum_j a_{ij} = \mu_i, \quad \sum_i a_{ij} = \nu_i, \tag{1}$$

と固定された分割表 $A = [a_{ij}]$ が生じる確率を

$$p(A|\mu, \nu, \Lambda) = \frac{1}{Z(\Lambda)} \prod_{i,j} \frac{\lambda_{ij}^{a_{ij}}}{a_{ij}!}, \quad Z(\Lambda) = \sum_A \prod_{i,j} \frac{\lambda_{ij}^{a_{ij}}}{a_{ij}!}$$

と定めることができる. ここで $Z(\Lambda)$ の定義和における $A = [a_{ij}]$ は条件(1)を満たす分割表全体を走る. この確率分布を周辺度数がすべて固定されている場合の分割表の確率分布と呼ぶことにする.

$$\phi_{kl} = \frac{\lambda_{kl}\lambda_{k+1,l+1}}{\lambda_{k+1,l}\lambda_{k,l+1}}, \quad s_{kl} = \sum_{i=1}^{k} \sum_{j=1}^{l} a_{ij}$$

とおくと,

$$a_{ij} = s_{ij} + s_{i-1,j-1} - s_{i-1,j} - s_{i,j-1}$$

なので, 上の確率は次のようにも書ける:

$$p(A|\mu, \nu, \Lambda) = \frac{1}{\widetilde{Z}(\Lambda)} \frac{\prod_{k,l} \phi_{kl}^{s_{kl}}}{\prod_{i,j} a_{ij}!}, \quad \widetilde{Z}(\Lambda) = \sum_A \frac{\prod_{k,l} \phi_{kl}^{s_{kl}}}{\prod_{i,j} a_{ij}!}.$$

ここで, i, j, k, l はそれぞれ $i = 1, \dots, r, j = 1, \dots, c, k = 1, \dots, r-1, l = 1, \dots, c-1$ を走り, A は条件(1)を満たす分割表全体を走る.

このとき, パラメーター $\Lambda = [\lambda_{ij}]$ の独立性は

$$\phi_{kl} = 1 \quad (k = 1, \dots, r-1, \ l = 1, \dots, c-1)$$

という $(r-1)(c-1)$ 個の連立条件で書ける. パラメーター $\Lambda = [\lambda_{ij}]$ が独立性を満たしているとき, λ_{ij} は

$$\lambda_{ij} = \frac{\mu_i \nu_j}{n}$$

に一意的に決まってしまい, 上の確率は次の形になる:

$$p(A|\mu, \nu, \Lambda) = \frac{\prod_i \mu_i! \cdot \prod_j \nu_j!}{n! \prod_{i,j} a_{ij}!}. \tag{2}$$

すなわち, パラメーター $\Lambda = [\lambda_{ij}]$ が独立性を満たしているとき,

$$\widetilde{Z}(\Lambda) = \frac{n!}{\prod_i \mu_i! \cdot \prod_i \nu_j!}$$

になる. 確率(2)は次のように書き直される:

$$p(A|\mu,\nu,\Lambda) = \frac{\displaystyle\prod_{j=1}^{c} \binom{\nu_j}{a_{1j},\dots,a_{rj}}}{\displaystyle\binom{n}{\mu_1,\dots,\mu_r}}. \tag{3}$$

ここで, 多項係数を次のように書いた:

$$\binom{n}{m_1,\dots,m_r} = \frac{n!}{m_1!\cdots m_r!}, \quad m_1+\cdots+m_r = n.$$

この多項係数は n 個のものを m_1 個, …, m_r 個に分割する方法の個数を表している. 確率(3)の分子分母は以下のような意味を持っている.

- 番号が j の玉が ν_j 個ある状況を考える.
- 全部で $\sum_j \nu_j = n$ 個の玉達の全体を m_1 個, …, m_r 個に分割する.
- (3)の分母は n 個の玉達の全体を m_1 個, …, m_r 個に分割する方法の個数になっている.
- (3)の分子の各因子は ν_j 個の番号 j の玉達を a_{1j} 個, …, a_{rj} 個に分割する方法の個数になっている.

このことから, (3)で定義される確率分布がどのようなものであるかがわかる.

周辺度数がすべて固定されている分割表の確率分布のパラメーター全体の空間の次元は $(r-1)(c-1)$ になり, 独立性の条件を満たすパラメーター達のなす部分空間の次元は 0 になり, 全体の次元との差は $(r-1)(c-1)$ になる.

注意: $r=c=2$ の場合の周辺度数がすべて固定されている 2×2 の分割表の確率分布は Fisher's noncentral hypergeometric distribution と呼ばれており, その独立性を満たす場合は hypergeometric distribution と呼ばれている. □

1.5 周辺度数がすべて固定されている分割表の独立性を満たす確率分布の漸近挙動

前節の記号をそのまま引き継ぎ,

$$\lambda_{ij} = np_{ij}, \quad a_{ij} - \lambda_{ij} = \sqrt{n}\, x_{ij}$$

となっていると仮定する. このとき, 前節の式(2)の中の階乗にStirlingの近似公式を適用すると, $n\to\infty$ において,

$$\begin{aligned}
p(A|\mu,\nu,\Lambda) &= \frac{\prod_i (\mu_i^{\mu_i} e^{-\mu_i}\sqrt{2\pi\mu_i})\cdot\prod_j(\nu_j^{\nu_j}e^{-\nu_j}\sqrt{2\pi\nu_j})!}{n^n e^{-n}\sqrt{2\pi n}\prod_{i,j}(a_{ij}^{a_{ij}}e^{-a_{ij}}\sqrt{2\pi a_{ij}})}(1+o(1)) \\
&= \frac{\prod_i (\mu_i^{\mu_i} \sqrt{2\pi\mu_i})\cdot\prod_j(\nu_j^{\nu_j}\sqrt{2\pi\nu_j})!}{n^n \sqrt{2\pi n}\prod_{i,j}(a_{ij}^{a_{ij}}\sqrt{2\pi a_{ij}})}(1+o(1)) \\
&= \sqrt{\frac{\prod_i \mu_i\cdot\prod_j\nu_j}{(2\pi)^{(r-1)(c-1)}\prod_{i,j}a_{ij}}}\exp\left(-\sum_{i,j}a_{ij}\log\frac{na_{ij}}{\mu_i\nu_j}\right)(1+o(1)) \\
&= \sqrt{\frac{\prod_i \mu_i\cdot\prod_j\nu_j}{(2\pi)^{(r-1)(c-1)}\prod_{i,j}a_{ij}}}\exp\left(-\sum_{i,j}a_{ij}\log\frac{a_{ij}}{\lambda_{ij}}\right)(1+o(1)) \\
&= \sqrt{\left(\frac{n}{2\pi}\right)^{(r-1)(c-1)}}\exp\left(-\sum_{i,j}a_{ij}\log\frac{a_{ij}}{\lambda_{ij}}\right)(1+o(1))
\end{aligned}$$

1つ目の等号でStirlingの公式を用い, 2つ目の等号で $\sum_i\mu_i = \sum_j\nu_j = \sum_{i,j}a_{ij} = n$ を使い, 3つ目の等号では指数部分に

$$\mu_i = \sum_j a_{ij}, \quad \nu_i = \sum_i a_{ij}, \quad n = \sum_{i,j}a_{ij}$$

を使い, 4つ目の等号では $\lambda_{ij} = \mu_i\nu_j/n$ を使い, 5つ目の等号では $a_{ij} = \lambda_{ij}(1+o(1))$ と $\lambda_{ij} = \mu_i\nu_j/n$ を使った.

このとき, λ を大きくすると, 上で得た近似式の指数函数の中身の -2 倍は次のように近似される:

$$
\begin{aligned}
2\sum_{i,j} a_{ij}\log\frac{a_{ij}}{\lambda_{ij}} &= 2\sum_{i,j}\lambda_{ij}\left(1+\frac{a_{ij}-\lambda_{ij}}{\lambda_{ij}}\right)\log\left(1+\frac{a_{ij}-\lambda_{ij}}{\lambda_{ij}}\right)\\
&= 2\sum_{i,j}\left(a_{ij}-\lambda_{ij}+\frac{(a_{ij}-\lambda_{ij})^2}{2\lambda_{ij}}\right)+O\left(\frac{1}{\sqrt{n}}\right)\\
&= \sum_{i,j}\frac{(a_{ij}-\lambda_{ij})^2}{\lambda_{ij}}+O\left(\frac{1}{\sqrt{n}}\right).
\end{aligned}
$$

1つ目の等号で $a_{ij}=\lambda_{ij}(1+(a_{ij}-\lambda_{ij})/\lambda_{ij})$ を用い, 2つ目の等号で $\log(1+x)=x-x^2/2+O(x^3)$ を用い, 3つ目の等号で $\sum_{i,j}a_{ij}=\sum_{i,j}\lambda_{ij}=n$ を用いた.

さらに, $\lambda_{ij}=np_{ij}$, $a_{ij}-\lambda_{ij}=\sqrt{n}\,x_{ij}$ より,

$$
\begin{aligned}
2\sum_{i,j} a_{ij}\log\frac{a_{ij}}{\lambda_{ij}} &= \sum_{i,j}\frac{(a_{ij}-\lambda_{ij})^2}{\lambda_{ij}}+O\left(\frac{1}{\sqrt{n}}\right)\\
&= \sum_{i,j}\frac{x_{ij}^2}{p_{ij}}+O\left(\frac{1}{\sqrt{n}}\right)
\end{aligned}
$$

でかつ $da_{ij}=\sqrt{n}\,dx_{ij}$ より,

$$
p(A|\mu,\nu,\Lambda)\prod_{i=1}^{r-1}\prod_{j=1}^{c-1}da_{ij}\approx\frac{1}{\sqrt{(2\pi)^{(r-1)(c-1)}}}\exp\left(-\frac{1}{2}\sum_{i,j}\frac{x_{ij}^2}{p_{ij}}\right)\prod_{i=1}^{r-1}\prod_{j=1}^{c-1}dx_{ij}.
$$

以上の計算結果から, 周辺度数がすべて固定されている分割表の独立性を満たす確率分布は, n が大きなときに, 台が $(r-1)(c-1)$ 次元の多変量正規分布で近似され, 統計量

$$
X^2=\sum_{i,j}\frac{(a_{ij}-\lambda_{ij})^2}{\lambda_{ij}}=\sum_{i,j}\frac{x_{ij}^2}{p_{ij}}
$$

が漸近的に自由度 $(r-1)(c-1)$ のχ²分布に従うことがわかる. 一般に台が N 次元の

$$
\text{const.}\exp\left(-\frac{1}{2}\sum_{i,j}a_{ij}x_ix_j\right)\times(\text{delta function with }N\text{-dim. support})
$$

の形の平均が 0 の多変量正規分布において, $\sum_{i,j}a_{ij}x_ix_j$ の部分に対応する統計量は自由度 N のχ²分布を満たすことを使った.

さらに, 上の X^2 で近似される統計量

$$
G=2\sum_{i,j}a_{ij}\log\frac{a_{ij}}{\lambda_{ij}}
$$

も漸近的に自由度 $(r-1)(c-1)$ のχ²分布に従うこともわかる.

後で周辺度数がすべて固定されていない場合にも同様の結果が得られることを説明する.

2 Pearsonのχ²統計量

2.1 Pearsonのχ²統計量とG統計量の定義

$r\times c$ の分割表 $A=[a_{ij}]$ に対して, 以下の量をPearsonのχ²統計量と呼ぶ:

$$
X^2=\sum_{i,j}\frac{(O_{ij}-E_{ij})^2}{E_{ij}}.
$$

ここで,

$$
O_{ij}=a_{ij},\quad E_{ij}=\frac{M_iN_j}{n},\quad n=\sum_{i,j}a_{ij},\quad M_i=\sum_j a_{ij},\quad N_j=\sum_i a_{ij}.
$$

次のように G 統計量を定義しておく:

$$G = 2\sum_{i,j} O_{ij} \log \frac{O_{ij}}{E_{ij}}.$$

G と X^2 は, Taylor展開

$$(1+x)\log(1+x) = (1+x)\left(x - \frac{x^2}{2} + O(x^3)\right) = x + \frac{x^2}{2} + O(x^3)$$

から得られる次の公式によって, 互いに相手を近似するという関係になっている:

$$2a \log \frac{a}{\lambda} = 2(a-\lambda) + \frac{(a-\lambda)^2}{\lambda} + O\left(\frac{(a-\lambda)^3}{\lambda^2}\right)$$

$\sum_{i,j} O_{ij} = \sum_{i,j} E_{ij}$ より, $a - \lambda = O_{ij} - E_{ij}$ の和が消えることに注意せよ.

2.1.1 実装と計算の例

In [1]:

```
safediv(x, y) = iszero(x) ? x : x/y
safemult(x, y) = iszero(x) ? x : x*y

function chisq(A)
    r, c = size(A)
    n = sum(A)
    M = vec(sum(A, dims=2))
    N = vec(sum(A, dims=1))
    sum(safediv((A[i,j] - M[i]*N[j]/n)^2, M[i]*N[j]/n) for i in 1:r, j in 1:c)
end

function gstat(A)
    r, c = size(A)
    n = sum(A)
    M = vec(sum(A, dims=2))
    N = vec(sum(A, dims=1))
    2sum(safemult(A[i,j], log(A[i,j]) - log(M[i]*N[j]/n)) for i in 1:r, j in 1:c)
end

A = [
    1 2 3
    2 4 6
]
@show chisq(A)
@show gstat(A)

B = [
    1 8 1
    2 1 6
]
@show chisq(B)
@show gstat(B);
```

```
chisq(A) = 0.0
gstat(A) = 0.0
chisq(B) = 9.322398589065257
gstat(B) = 10.447245765410694
```

2.1.2 2×2の場合のPearsonのχ²統計量の具体形

2×2 の分割表

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

のPearsonのχ²統計量は

$$X^2 = \frac{(ad-bc)^2(a+b+c+d)}{(a+b)(a+c)(b+d)(c+d)}$$

と表わされる.

In [2]:
```
using SymPy

@vars a b c d
A = [
    a b
    c d
]
chisq(A).factor()
```

Out[2]:
$$\frac{(ad-bc)^2\,(a+b+c+d)}{(a+b)\,(a+c)\,(b+d)\,(c+d)}$$

2.2 分割表におけるPearsonのχ²統計量が漸近的に満たす確率分布

定理: 分割表について前節で定義した4つの確率分布のどれにおいても, そのパラメーターが独立性を満たしているならば, Pearsonのχ²統計量 X^2 と G 統計量 G はともに, λ もしくは n を大きくするとき漸近的に, 自由度 $(r-1)(c-1)$ のχ²分布に従う. □

前節において, 周辺度数がすべて固定されている分割表の独立性を満たす確率分布の場合にはこれが成立することをすでに示した.

他の3つの場合のこの定理はWilksの定理から導かれる. 以下ではこの定理の成立を数値的に確認してみよう.

2.3 3×4 の場合の数値的確認

In [3]:
```
using Distributions
using Plots

# Plots.jlのデフォルトの設定を表示
#@show Plots.reset_defaults()

# legendの半透明化
@show default(:bglegend, plot_color(default(:bg), 0.5))
@show default(:fglegend, plot_color(ifelse(isdark(plot_color(default(:bg))), :white, :black), 0.6));

using Base64
displayfile(mime, file; tag="img") = open(file) do f
    base64 = base64encode(f)
    display("text/html", """<$(tag) src="data:$(mime);base64,$(base64)"/>""")
end

pyplotclf() = if backend() == Plots.PyPlotBackend(); PyPlot.clf(); end

pyplot(fmt=:svg)
```

```
default(:bglegend, plot_color(default(:bg), 0.5)) = RGBA{Float64}(1.0,1.0,1.0,0.5)
default(:fglegend, plot_color(ifelse(isdark(plot_color(default(:bg))), :white, :black), 0.6)) = RGBA{Float64}(0.0,0.
0,0.0,0.6)
```

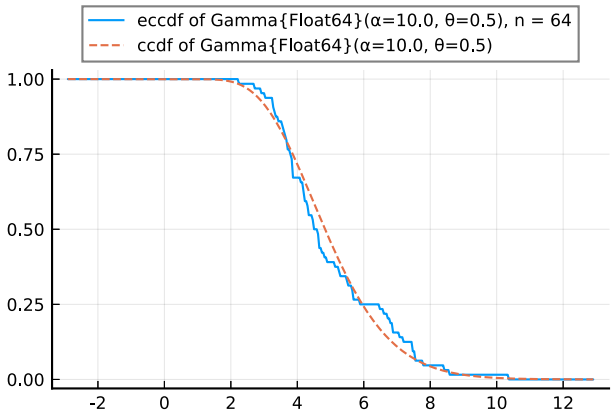
Out[3]:
```
Plots.PyPlotBackend()
```

In [4]:

```
ecdf(x; Y=rand(10)) = mean(Y .≤ x)
eccdf(x; Y=rand(10)) = mean(Y .≥ x)

n = 2^6
dist = Gamma(10, 0.5)
Y = rand(dist, n)
x = range(mean(dist)-5std(dist), mean(dist)+5std(dist), length=400)
plot(size=(400, 270), legend=:outertop)
plot!(x, eccdf.(x; Y=Y); label="eccdf of $dist, n = $n")
plot!(x, ccdf.(dist, x); label="ccdf of $dist", ls=:dash)
```

Out[4]:

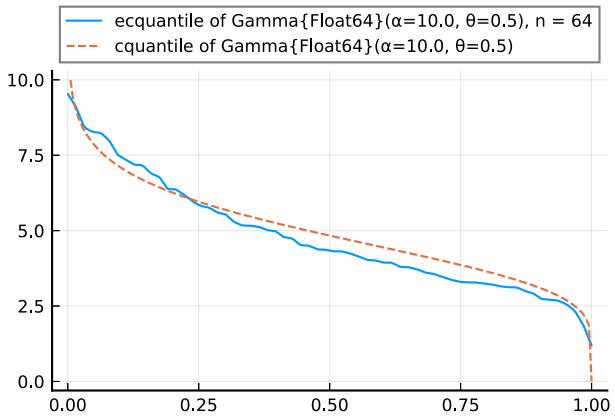


In [5]:

```
equantile(p; Y=randn(100)) = quantile(Y, p)
ecquantile(p; Y=randn(100)) = quantile(Y, 1-p)

n = 2^6
dist = Gamma(10, 0.5)
Y = rand(dist, n)
p = range(0, 1, length=200)
plot(size=(400, 270), legend=:outertop)
plot!(p, ecquantile.(p; Y=Y); label="ecquantile of $dist, n = $n")
plot!(p, cquantile.(dist, p); label="cquantile of $dist", ls=:dash)
```

Out[5]:



In [6]:

```
param_indep(p, q) = p .* q'
p = [0.2, 0.3, 0.5]
q = [0.1, 0.2, 0.3, 0.4]
P = param_indep(p, q)
@show p
@show q
@show sum(P)
P
```

```
p = [0.2, 0.3, 0.5]
q = [0.1, 0.2, 0.3, 0.4]
sum(P) = 1.0000000000000002
```

Out[6]:

```
3×4 Array{Float64,2}:
 0.02  0.04  0.06  0.08
 0.03  0.06  0.09  0.12
 0.05  0.1   0.15  0.2
```

/

In [7]:

```
df_chisq(r, c) = (r-1)*(c-1)
df_chisq(P) = prod(size(P) .- 1)
@show size(P)
@show size(P) .- 1
@show df_chisq(P);
```

```
size(P) = (3, 4)
size(P) .- 1 = (2, 3)
df_chisq(P) = 6
```

2.3.1 数値的確認: rc 個のPoisson分布の直積の場合

In [8]:

```
function plot_sim(title, PearsonChisq, G_Statistics, df, binstep)
    chisq_dist = Chisq(df)
    f(x) = pdf(chisq_dist, x)
    xmax = 4df + 2
    x = range(0, xmax, length=200)
    bin = range(0, xmax, step=binstep)

    P1 = plot(xlabel="x")
    plot!(title=title, titlefontsize=9, legendfontsize=8, guidefontsize=8)
    histogram!(PearsonChisq; bin=bin, norm=true, alpha=0.3, label="Pearson's χ²-statistics")
    plot!(x, f.(x); label="pdf of Chisq(df=$(df))")
    plot!(tickfontsize=7)

    P2 = plot(xlabel="x")
    plot!(title=title, titlefontsize=9, legendfontsize=8, guidefontsize=8)
    histogram!(G_Statistics; bin=bin, norm=true, alpha=0.3, label="G-statistics")
    plot!(x, f.(x); label="pdf of Chisq(df=$(df))")
    plot!(tickfontsize=7)

    P3 = plot(guidefontsize=8)
    plot!(xlabel="ccdf of Chisq(df=$(df))", ylabel="eccdf of Pearson's χ²-statistics")
    xx = ccdf.(chisq_dist, x)
    yy = eccdf.(x; Y=PearsonChisq)
    plot!(xx, yy; label="")
    plot!([0,1], [0,1]; label="", color=:black, ls=:dot, alpha=0.5)
    plot!(xtick=0:0.1:1, ytick=0:0.1:1, tickfontsize=7, xrotation=90)

    P4 = plot(guidefontsize=8)
    plot!(xlabel="ccdf of Chisq(df=$(df))", ylabel="eccdf of G-statistics")
    xx = ccdf.(chisq_dist, x)
    yy = eccdf.(x; Y=G_Statistics)
    plot!(xx, yy; label="")
    plot!([0,1], [0,1]; label="", color=:black, ls=:dot, alpha=0.7)
    plot!(xtick=0:0.1:1, ytick=0:0.1:1, tickfontsize=7, xrotation=90)

    α_max = 0.055
    x0 = range(cquantile(chisq_dist, α_max), 2xmax, length=200)

    P5 = plot(guidefontsize=8)
    plot!(xlabel="ccdf of Chisq(df=$(df))", ylabel="eccdf of Pearson's χ²-statistics")
    xx = ccdf.(chisq_dist, x0)
    yy = eccdf.(x0; Y=PearsonChisq)
    plot!(xx, yy; label="")
    plot!([0, α_max], [0, α_max]; label="", color=:black, ls=:dot, alpha=0.5)
    plot!(xtick=0:0.005:1, ytick=0:0.005:1, tickfontsize=7, xrotation=90)

    P6 = plot(guidefontsize=8)
    plot!(xlabel="ccdf of Chisq(df=$(df))", ylabel="eccdf of G-statistics")
    xx = ccdf.(chisq_dist, x0)
    yy = eccdf.(x0; Y=G_Statistics)
    plot!(xx, yy; label="")
    plot!([0, α_max], [0, α_max]; label="", color=:black, ls=:dot, alpha=0.5)
    plot!(xtick=0:0.005:1, ytick=0:0.005:1, tickfontsize=7, xrotation=90)

    plot(P1, P3, P5, P2, P4, P6;
        size=(800, 500), layout=grid(2, 3; widths=[3.2/8, 2.4/8, 2.4/8])
    )
end
```

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

In [9]:

```
prod_Poisson(Λ) = product_distribution(Poisson.(vec(Λ)))

function sim_Poisson(; λ=100, P=param_indep([0.2, 0.3, 0.5], [0.1, 0.2, 0.3, 0.4]), L=10^5)
    dist = prod_Poisson(λ*P)
    PearsonChisq = Array{Float64,1}(undef, L)
    G_Statistics = Array{Float64,1}(undef, L)
    for l in 1:L
        A = reshape(rand(dist), size(P))
        PearsonChisq[l] = chisq(A)
        G_Statistics[l] = gstat(A)
    end
    PearsonChisq, G_Statistics
end

function plot_sim_Poisson(; λ=100, P=param_indep([0.2, 0.3, 0.5], [0.1, 0.2, 0.3, 0.4]),
        L=10^5, binstep=0.5
    )
    @show expectation = λ*P
    @show total = sum(expectation)
    @show r, c = size(expectation)
    @show df = df_chisq(expectation)
    @time PearsonChisq, G_Statistics =  sim_Poisson(λ=λ, P=P, L=L)
    title = "$(r)×$(c) Poisson distributions (λ = $(λ))"
    sleep(0.1)

    plot_sim(title, PearsonChisq, G_Statistics, df, binstep)
end
```

Out[9]:

```
plot_sim_Poisson (generic function with 1 method)
```

In [10]:

```
P = param_indep([0.2, 0.3, 0.5], [0.1, 0.2, 0.3, 0.4])
```

Out[10]:

```
3×4 Array{Float64,2}:
 0.02  0.04  0.06  0.08
 0.03  0.06  0.09  0.12
 0.05  0.1   0.15  0.2
```

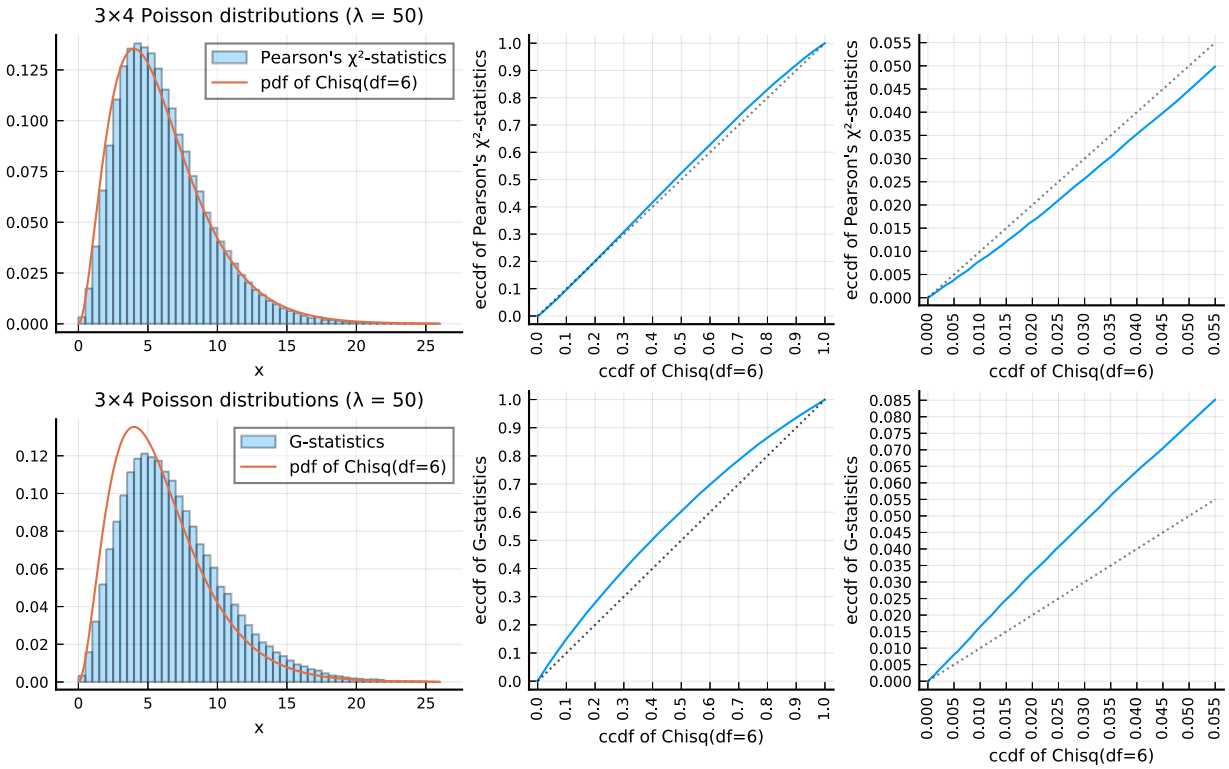
/

In [11]:

```
1  plot_sim_Poisson(λ=50, P=P)
```

```
expectation = λ * P = [1.0000000000000002 2.0000000000000004 3.0 4.000000000000001; 1.5 3.0 4.5 6.0; 2.5 5.0 7.5 10.
0]
total = sum(expectation) = 50.0
(r, c) = size(expectation) = (3, 4)
df = df_chisq(expectation) = 6
  1.290666 seconds (3.10 M allocations: 149.537 MiB, 3.67% gc time)
```

Out[11]:



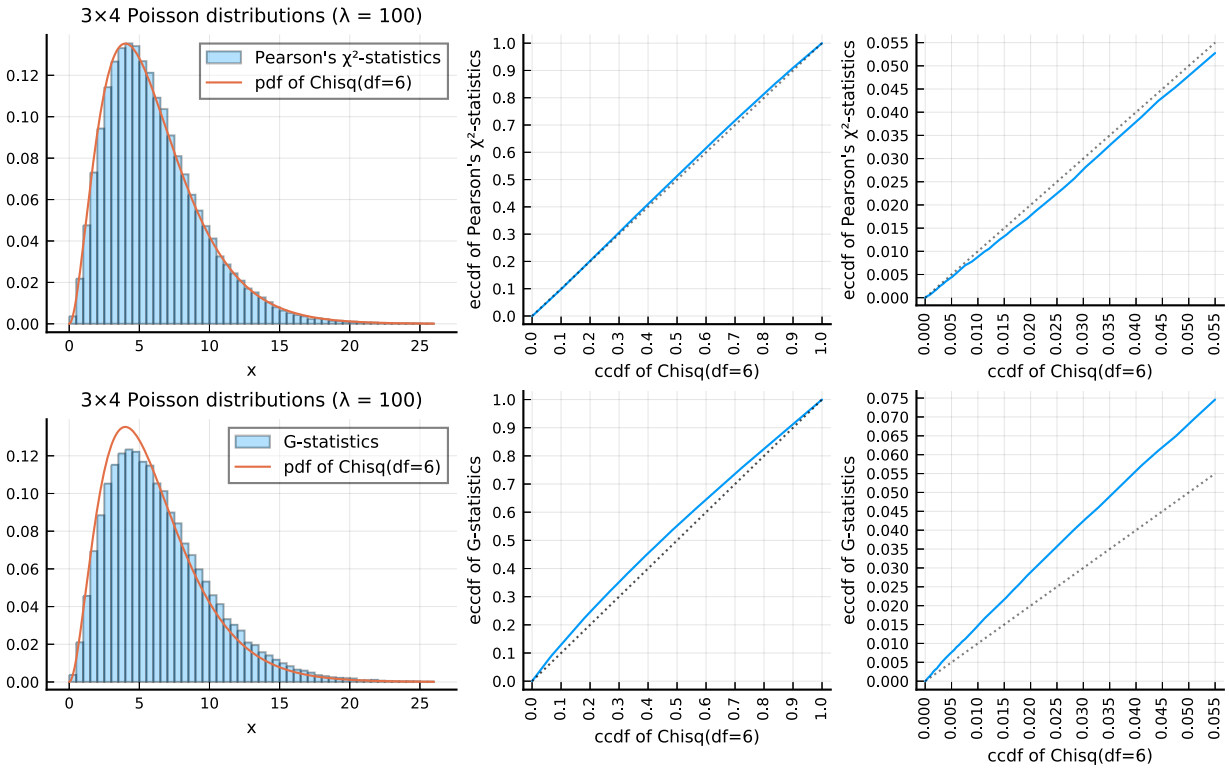
/

In [12]:

```
1  plot_sim_Poisson(λ=100, P=P)
```

```
expectation = λ * P = [2.0000000000000004 4.000000000000001 6.0 8.000000000000002; 3.0 6.0 9.0 12.0; 5.0 10.0 15.0 2
0.0]
total = sum(expectation) = 100.0
(r, c) = size(expectation) = (3, 4)
df = df_chisq(expectation) = 6
  1.099794 seconds (3.10 M allocations: 149.537 MiB, 3.80% gc time)
```

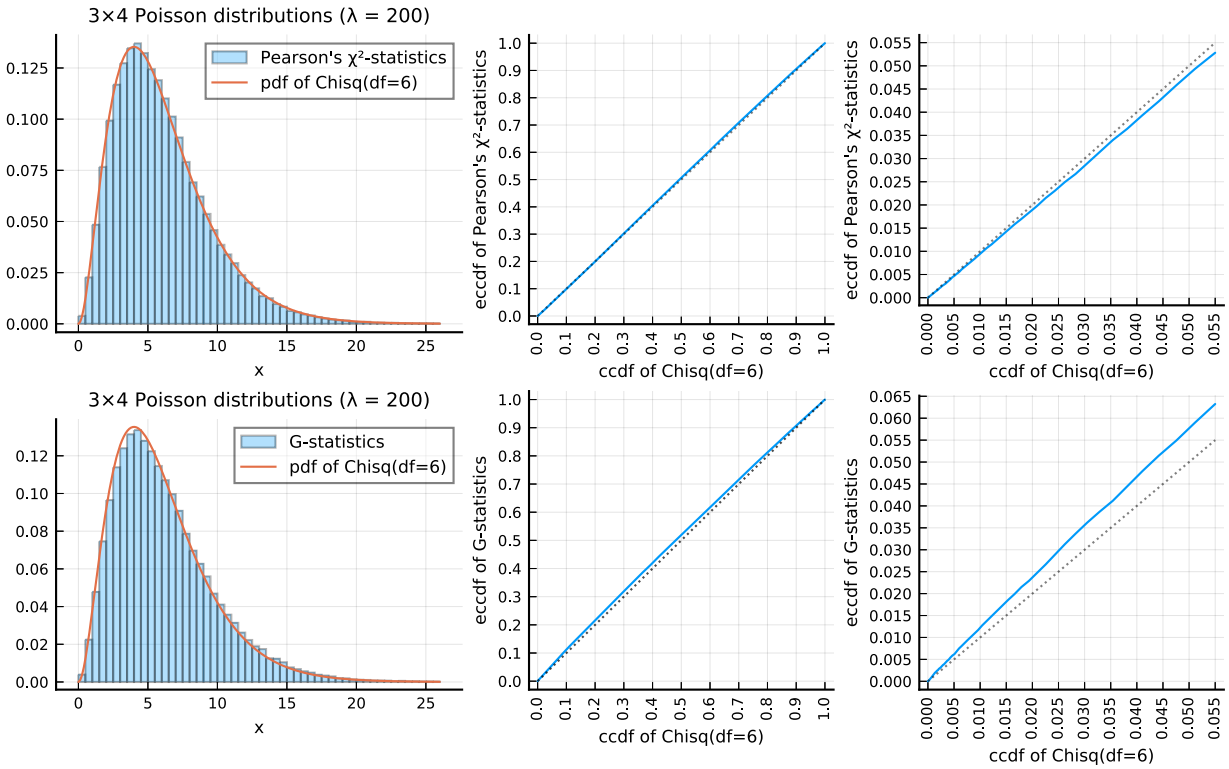
Out[12]:



In [13]:
```
plot_sim_Poisson(λ=200, P=P)
```

```
expectation = λ * P = [4.000000000000001 8.000000000000002 12.0 16.000000000000004; 6.0 12.0 18.0 24.0; 10.0 20.0 30.0 40.0]
total = sum(expectation) = 200.0
(r, c) = size(expectation) = (3, 4)
df = df_chisq(expectation) = 6
  1.265358 seconds (3.10 M allocations: 149.537 MiB, 4.13% gc time)
```

Out[13]:



小さな λ での誤差は G 統計量よりも, Pearsonのχ^2統計量の方が小さい.

2.3.2 数値的確認: rc 項分布の場合

In [14]:
```
function sim_Multinomial(; n=100, P=param_indep([0.2, 0.3, 0.5], [0.1, 0.2, 0.3, 0.4]), L=10^5)
    dist = Multinomial(n, vec(P))
    PearsonChisq = Array{Float64,1}(undef, L)
    G_Statistics = Array{Float64,1}(undef, L)
    for l in 1:L
        A = reshape(rand(dist), size(P))
        PearsonChisq[l] = chisq(A)
        G_Statistics[l] = gstat(A)
    end
    PearsonChisq, G_Statistics
end

function plot_sim_Multinomial(; n=100, P=param_indep([0.2, 0.3, 0.5], [0.1, 0.2, 0.3, 0.4]),
        L=10^5, binstep=0.5
    )
    @show expectation = n*P
    @show total = sum(expectation)
    @show r, c = size(expectation)
    @show df = df_chisq(expectation)
    @time PearsonChisq, G_Statistics =  sim_Multinomial(n=n, P=P, L=L)
    title = "$(r)×$(c)-nomial distribution (n = $(n))"
    sleep(0.1)

    plot_sim(title, PearsonChisq, G_Statistics, df, binstep)
end
```

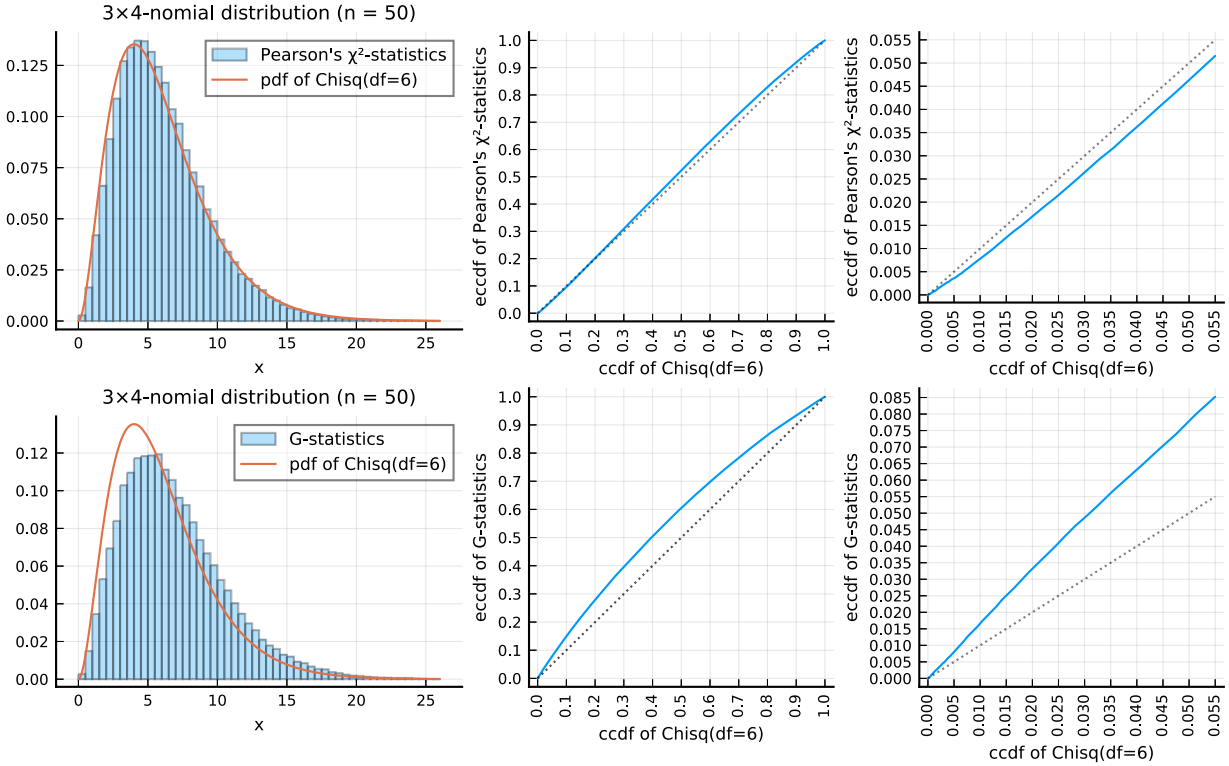
Out[14]:
```
plot_sim_Multinomial (generic function with 1 method)
```

In [15]:

```
1  plot_sim_Multinomial(n=50, P=P)
```

```
expectation = n * P = [1.0000000000000002 2.0000000000000004 3.0 4.000000000000001; 1.5 3.0 4.5 6.0; 2.5 5.0 7.5 10.
0]
total = sum(expectation) = 50.0
(r, c) = size(expectation) = (3, 4)
df = df_chisq(expectation) = 6
  0.562773 seconds (3.10 M allocations: 149.536 MiB, 8.50% gc time)
```

Out[15]:



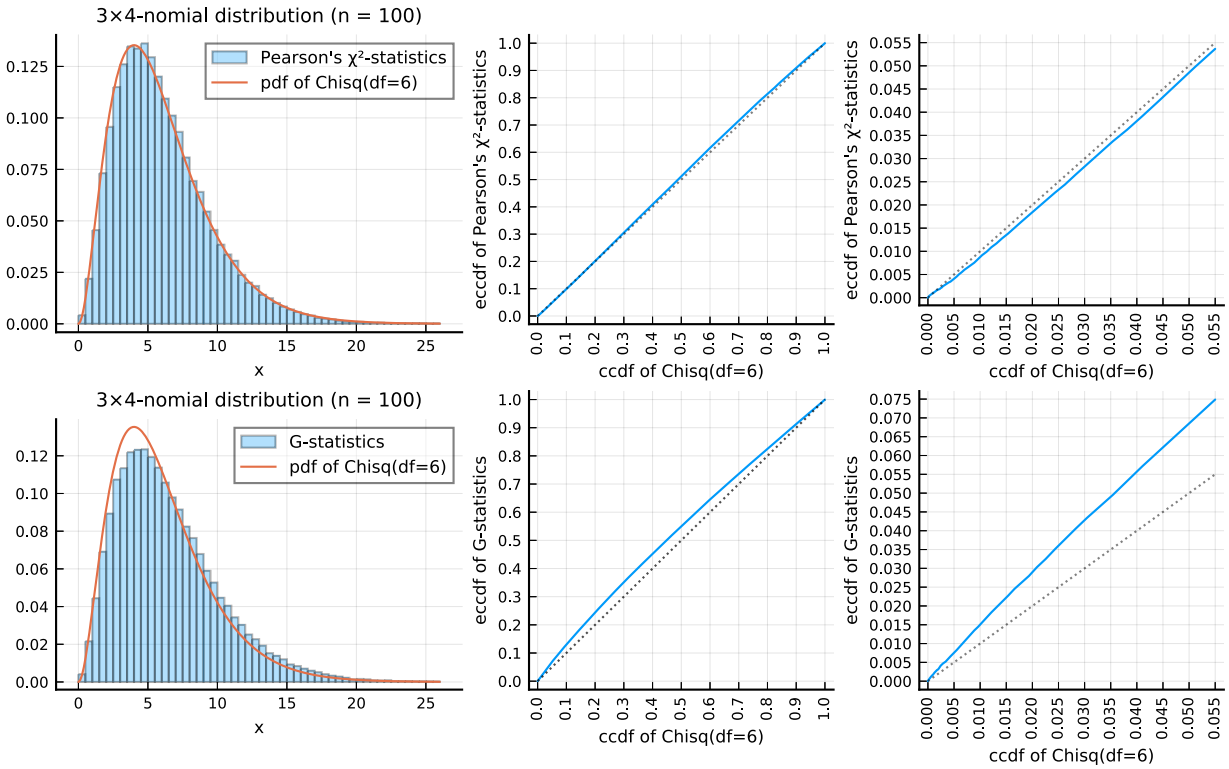
/

In [16]:

```
1  plot_sim_Multinomial(n=100, P=P)
```

```
expectation = n * P = [2.0000000000000004 4.000000000000001 6.0 8.000000000000002; 3.0 6.0 9.0 12.0; 5.0 10.0 15.0 2
0.0]
total = sum(expectation) = 100.0
(r, c) = size(expectation) = (3, 4)
df = df_chisq(expectation) = 6
  0.682386 seconds (3.10 M allocations: 149.536 MiB, 7.32% gc time)
```

Out[16]:

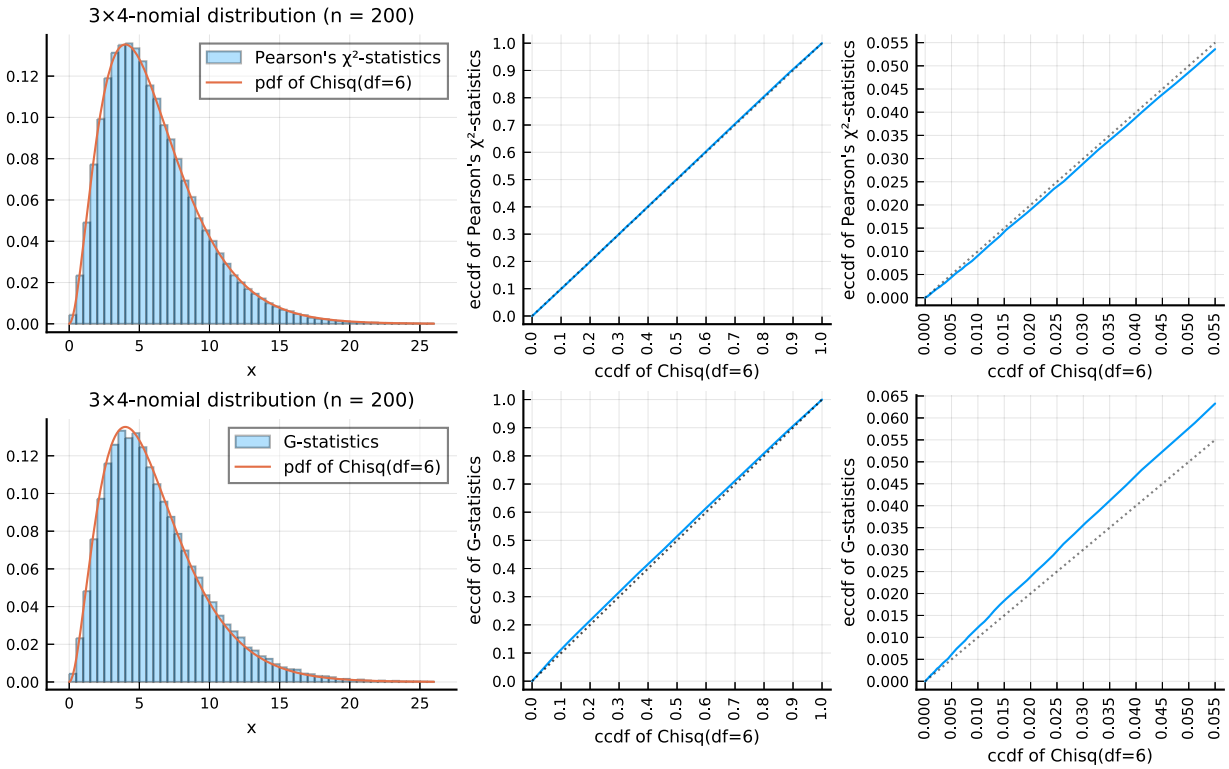


/

In [17]:
```
1  plot_sim_Multinomial(n=200, P=P)
```

```
expectation = n * P = [4.000000000000001 8.000000000000002 12.0 16.000000000000004; 6.0 12.0 18.0 24.0; 10.0 20.0 30.
0 40.0]
total = sum(expectation) = 200.0
(r, c) = size(expectation) = (3, 4)
df = df_chisq(expectation) = 6
  0.649971 seconds (3.10 M allocations: 149.536 MiB, 8.89% gc time)
```

Out[17]:



小さな n での誤差は G 統計量よりも, Pearsonのχ²統計量の方が小さい.

2.3.3 数値的確認: r 個の c 項分布の直積分布の場合

In [18]:
```
1  function rand_prod_Multinomial(M, q)
2      r, c = length(M), length(q)
3      A = Array{Int, 2}(undef, r, c)
4      for i in 1:r
5          A[i,:] = rand(Multinomial(M[i], q))
6      end
7      A
8  end
```

Out[18]:
```
rand_prod_Multinomial (generic function with 1 method)
```

In [19]:
```
1  M = [20, 30, 50]
2  q = [0.1, 0.2, 0.3, 0.4]
3  rand_prod_Multinomial(M, q)
```

Out[19]:
```
3×4 Array{Int64,2}:
 2  3   4  11
 3  5   8  14
 2  7  17  24
```

In [20]:

```
function sim_prod_Multinomial(; M=[20, 30, 50], q=[0.1, 0.2, 0.3, 0.4], L=10^5)
    PearsonChisq = Array{Float64,1}(undef, L)
    G_Statistics = Array{Float64,1}(undef, L)
    for l in 1:L
        A = rand_prod_Multinomial(M, q)
        PearsonChisq[l] = chisq(A)
        G_Statistics[l] = gstat(A)
    end
    PearsonChisq, G_Statistics
end

function plot_sim_prod_Multinomial(; M=[20, 30, 50], q=[0.1, 0.2, 0.3, 0.4],
        L=10^5, binstep=0.5
    )
    n = sum(M)
    @show expectation = M*q'
    @show total = sum(expectation)
    @show r, c = size(expectation)
    @show df = df_chisq(expectation)
    @time PearsonChisq, G_Statistics =  sim_prod_Multinomial(M=M, q=q, L=L)
    if c == 2
        title = "$(r) binomial distributions (n = $n)"
    elseif c == 3
        title = "$(r) trinomial distributions (n = $n)"
    elseif c == 4
        title = "$(r) quadranomial distributions (n = $n)"
    else
        title = "$(r)  $(c)-nomial distributions (n = $n)"
    end
    sleep(0.1)

    plot_sim(title, PearsonChisq, G_Statistics, df, binstep)
end
```

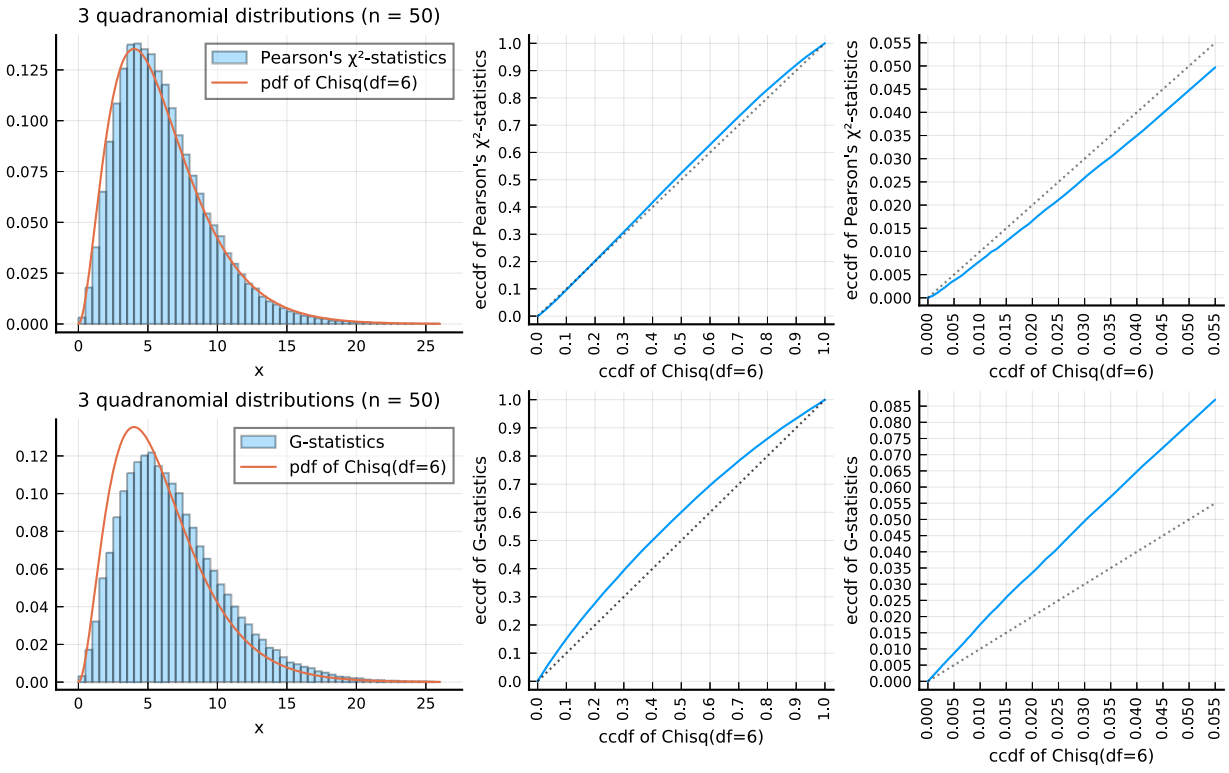
Out[20]: plot_sim_prod_Multinomial (generic function with 1 method)

In [21]:

```
plot_sim_prod_Multinomial(M=div.(M,2), q=q)
```

```
expectation = M * q' = [1.0 2.0 3.0 4.0; 1.5 3.0 4.5 6.0; 2.5 5.0 7.5 10.0]
total = sum(expectation) = 50.0
(r, c) = size(expectation) = (3, 4)
df = df_chisq(expectation) = 6
  0.746881 seconds (3.20 M allocations: 172.424 MiB, 6.70% gc time)
```

Out[21]:

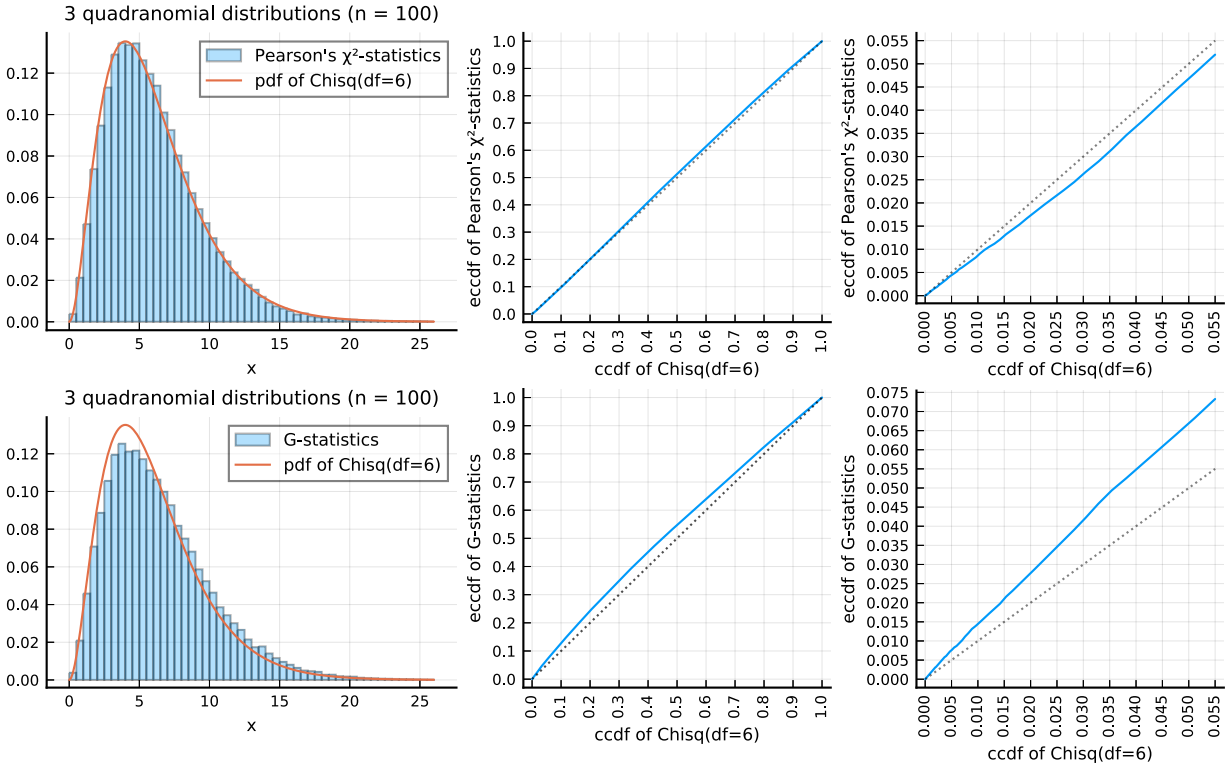


/

In [22]:
```
1  plot_sim_prod_Multinomial(M=M, q=q)
```

```
expectation = M * q' = [2.0 4.0 6.0 8.0; 3.0 6.0 9.0 12.0; 5.0 10.0 15.0 20.0]
total = sum(expectation) = 100.0
(r, c) = size(expectation) = (3, 4)
df = df_chisq(expectation) = 6
  0.706654 seconds (3.20 M allocations: 172.424 MiB, 6.82% gc time)
```

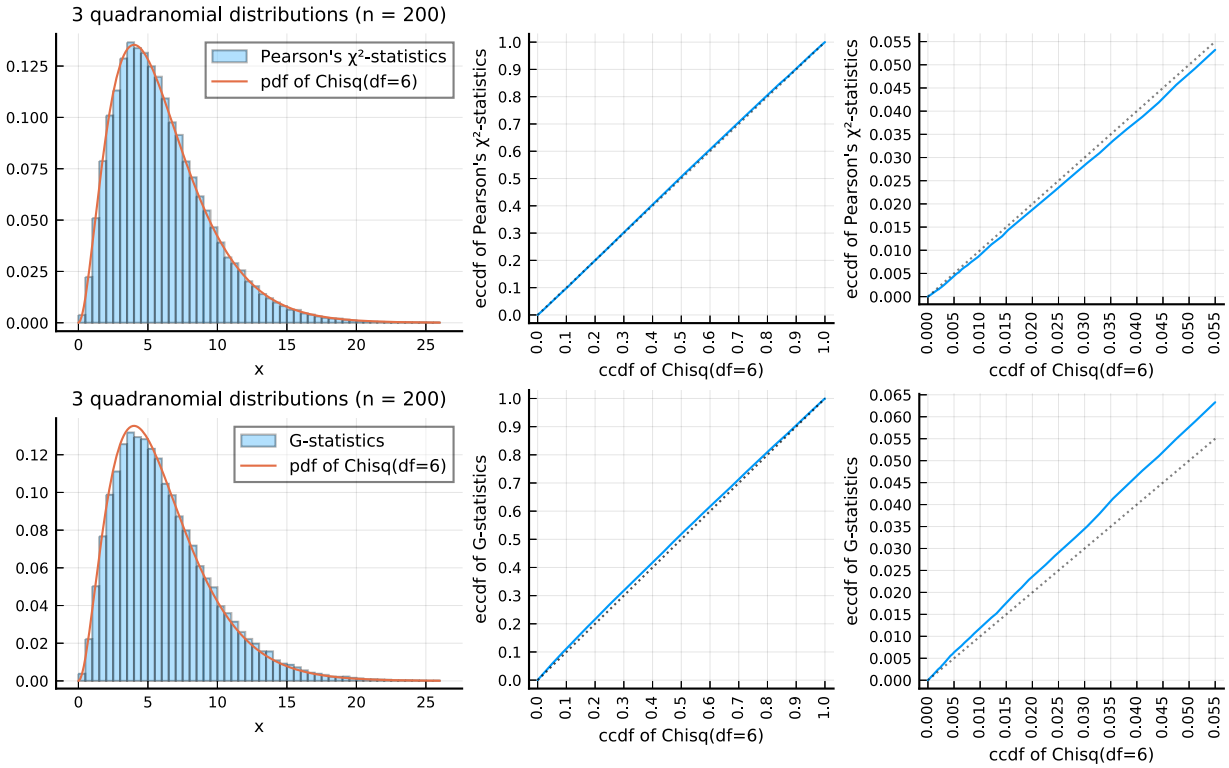
Out[22]:



In [23]:
```
1  plot_sim_prod_Multinomial(M=2M, q=q)
```

```
expectation = M * q' = [4.0 8.0 12.0 16.0; 6.0 12.0 18.0 24.0; 10.0 20.0 30.0 40.0]
total = sum(expectation) = 200.0
(r, c) = size(expectation) = (3, 4)
df = df_chisq(expectation) = 6
  0.570665 seconds (3.20 M allocations: 172.424 MiB, 7.11% gc time)
```

Out[23]:



以上のように, 周辺度数をすべて固定するという不自然な前提を採用しなくても, Pearsonのχ²統計量と G 統計量は漸近的に自由度

$(r-1)(c-1)$ のχ²分布に従っていることを数値的に確認できる.

そして, 小さな n での誤差は G 統計量よりも, Pearsonのχ²統計量の方が小さいことも確認できる. Pearsonのχ²統計量は優れた統計量である.

45度線に近いほど誤差が小さい. グラフが45度線よりも上にある部分はP値が余計に小さくなって有意差が出易くなることを意味している. 以上の例において, G 統計量は有意差が出易くなる方向で誤差が大きくなっている.

2.4 aoki-takemura-ohp.pdf の場合

aoki-takemura-ohp.pdf (http://www.math.kobe-u.ac.jp/crest-c/2009-09/Lec/aoki-takemura-ohp.pdf) の p.48 の例.

In [24]:
```
A = [
    2 1 1 0 0
    8 3 3 0 0
    0 2 1 1 1
    0 0 0 1 1
    0 0 0 0 1
]

r, c = size(A)
n = sum(A)
M = vec(sum(A, dims=2))
p = M/n
N = vec(sum(A, dims=1))
q = N/n
P = param_indep(p, q)
df = df_chisq(P)

@show chi_squared = chisq(A)
@show p_value = ccdf(Chisq(df), chi_squared)
n*P
```

```
chi_squared = chisq(A) = 25.337619047619047
p_value = ccdf(Chisq(df), chi_squared) = 0.06409042450667916
```

Out[24]:
```
5×5 Array{Float64,2}:
 1.53846   0.923077  0.769231  0.307692   0.461538
 5.38462   3.23077   2.69231   1.07692    1.61538
 1.92308   1.15385   0.961538  0.384615   0.576923
 0.769231  0.461538  0.384615  0.153846   0.230769
 0.384615  0.230769  0.192308  0.0769231  0.115385
```

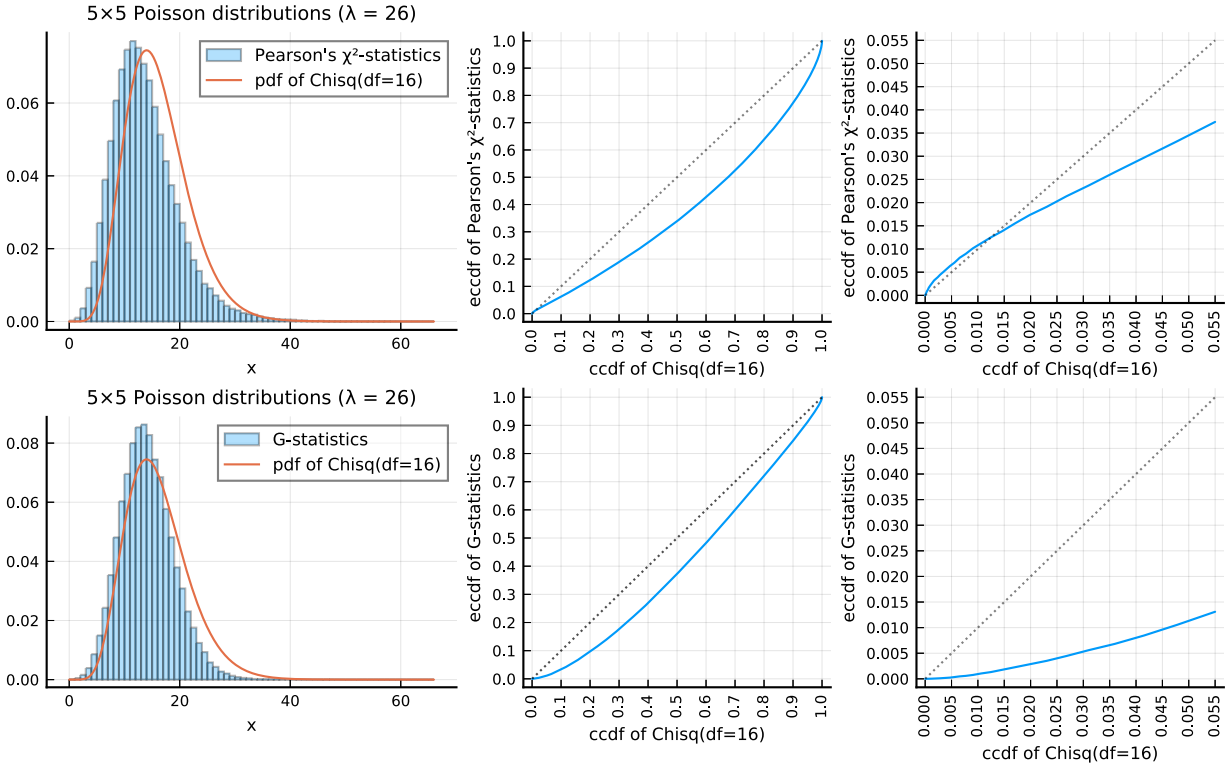
2.4.1 オリジナルの n=26 の場合

In [25]:

```
1  plot_sim_Poisson(λ=n, P=P; binstep=1)
```

```
expectation = λ * P = [1.5384615384615385 0.9230769230769231 0.7692307692307693 0.3076923076923077 0.4615384615384615
6; 5.384615384615385 3.230769230769231 2.6923076923076925 1.076923076923077 1.6153846153846154; 1.9230769230769231 1.
153846153846154 0.9615384615384616 0.38461538461538464 0.576923076923077; 0.7692307692307693 0.46153846153846156 0.38
461538461538464 0.15384615384615385 0.23076923076923078; 0.38461538461538464 0.23076923076923078 0.19230769230769232
0.07692307692307693 0.11538461538461539]
total = sum(expectation) = 26.0
(r, c) = size(expectation) = (5, 5)
df = df_chisq(expectation) = 16
  1.932493 seconds (3.10 M allocations: 166.322 MiB, 2.53% gc time)
```

Out[25]:



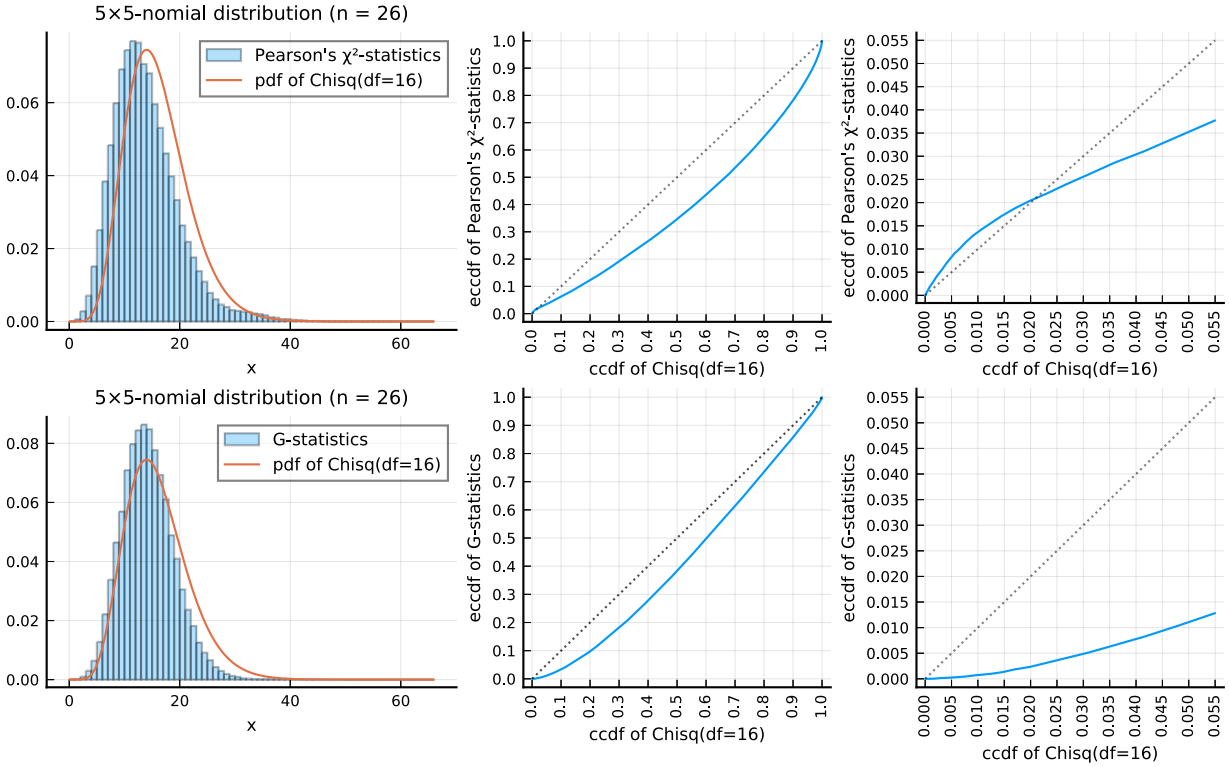
/

In [26]:

```
1  plot_sim_Multinomial(n=n, P=P; binstep=1)
```

```
expectation = n * P = [1.5384615384615385 0.9230769230769231 0.7692307692307693 0.3076923076923077 0.4615384615384615
6; 5.384615384615385 3.230769230769231 2.6923076923076925 1.076923076923077 1.6153846153846154; 1.9230769230769231 1.
153846153846154 0.9615384615384616 0.38461538461538464 0.576923076923077; 0.7692307692307693 0.46153846153846156 0.38
461538461538464 0.15384615384615385 0.23076923076923078; 0.38461538461538464 0.23076923076923078 0.19230769230769232
0.07692307692307693 0.11538461538461539]
total = sum(expectation) = 26.0
(r, c) = size(expectation) = (5, 5)
df = df_chisq(expectation) = 16
  0.913220 seconds (3.10 M allocations: 166.321 MiB, 5.20% gc time)
```

Out[26]:

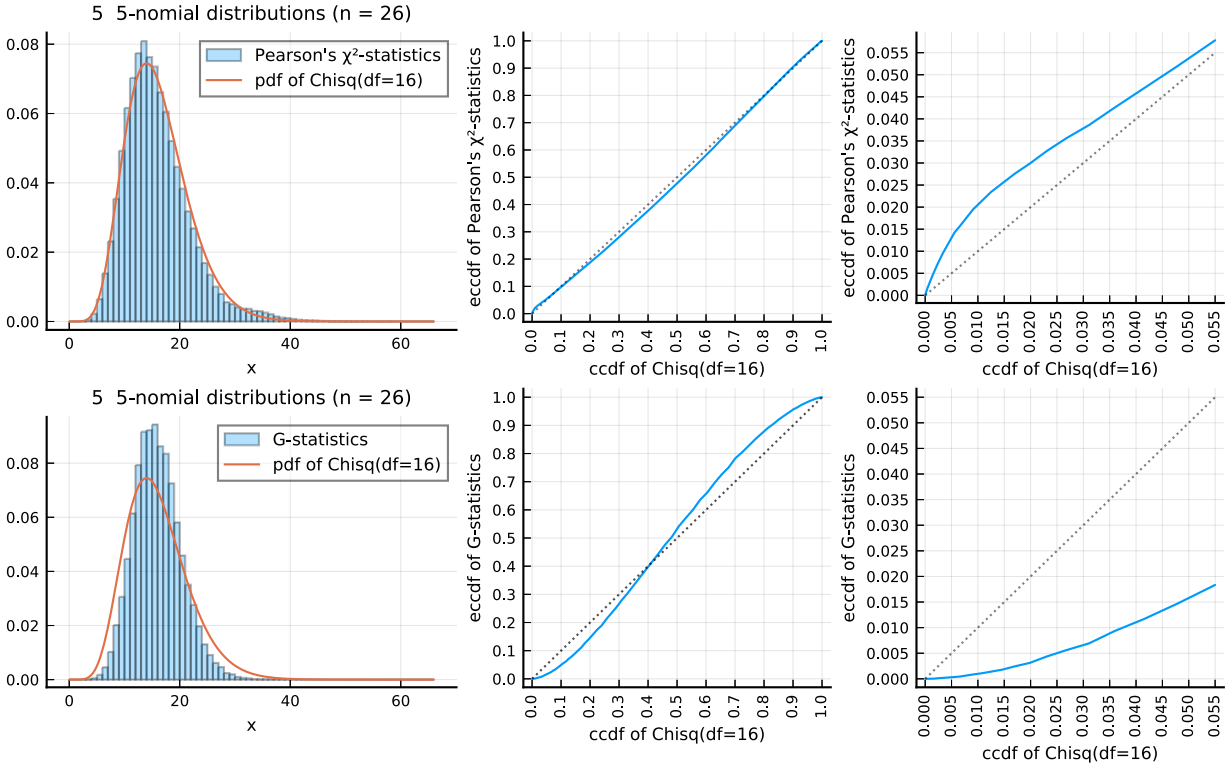


In [27]:

```
1  plot_sim_prod_Multinomial(M=M, q=q; binstep=1)
```

```
expectation = M * q' = [1.5384615384615385 0.9230769230769231 0.7692307692307693 0.3076923076923077 0.461538461538461
56; 5.384615384615385 3.230769230769231 2.6923076923076925 1.076923076923077 1.6153846153846154; 1.9230769230769231
1.153846153846154 0.9615384615384616 0.38461538461538464 0.576923076923077; 0.7692307692307693 0.46153846153846156 0.
38461538461538464 0.15384615384615385 0.23076923076923078; 0.38461538461538464 0.23076923076923078 0.1923076923076923
2 0.07692307692307693 0.11538461538461539]
total = sum(expectation) = 26.0
(r, c) = size(expectation) = (5, 5)
df = df_chisq(expectation) = 16
  0.801838 seconds (3.40 M allocations: 218.201 MiB, 5.74% gc time)
```

Out[27]:

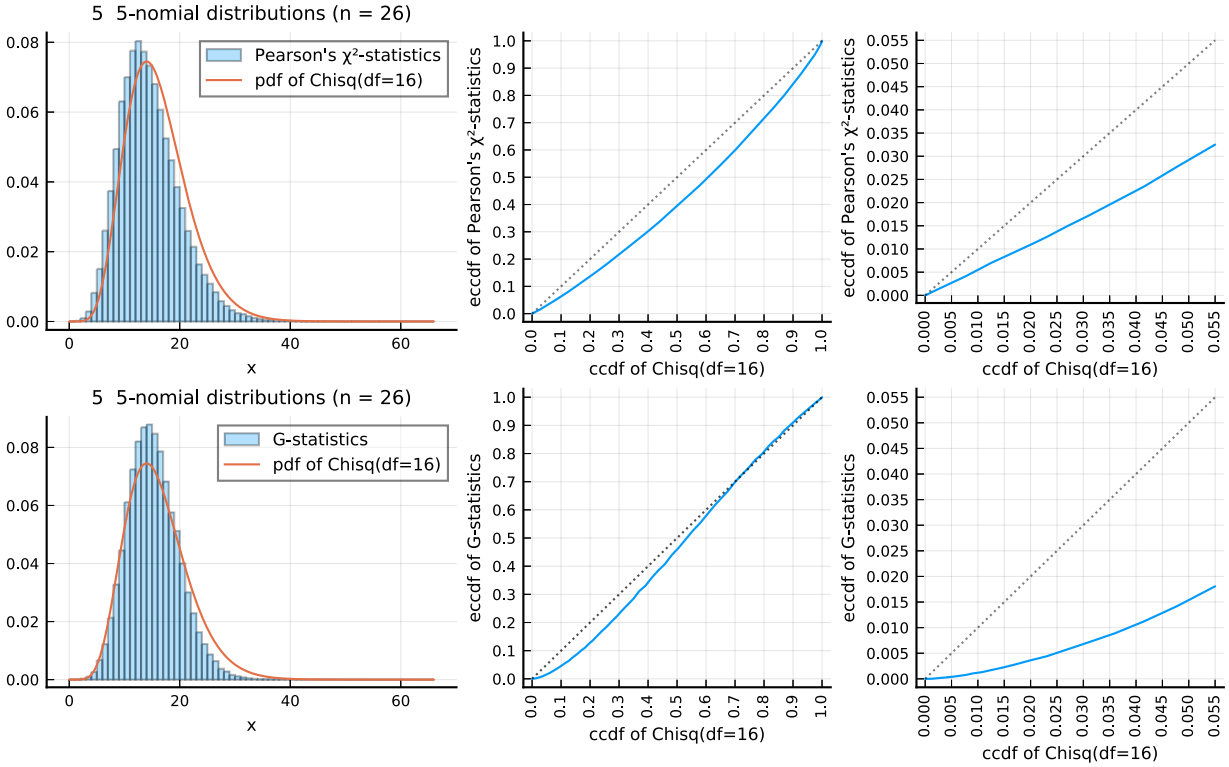


/

In [28]:
```
1  plot_sim_prod_Multinomial(M=N, q=p; binstep=1)
```

```
expectation = M * q' = [1.5384615384615385 5.384615384615384 1.9230769230769231 0.7692307692307693 0.3846153846153846
4; 0.9230769230769231 3.230769230769231 1.153846153846154 0.46153846153846156 0.23076923076923078; 0.7692307692307693
2.692307692307692 0.9615384615384616 0.38461538461538464 0.19230769230769232; 0.3076923076923077 1.0769230769230769
0.38461538461538464 0.15384615384615385 0.07692307692307693; 0.46153846153846156 1.6153846153846154 0.576923076923077
0.23076923076923078 0.11538461538461539]
total = sum(expectation) = 26.0
(r, c) = size(expectation) = (5, 5)
df = df_chisq(expectation) = 16
  0.790298 seconds (3.40 M allocations: 218.201 MiB, 11.79% gc time)
```

Out[28]:



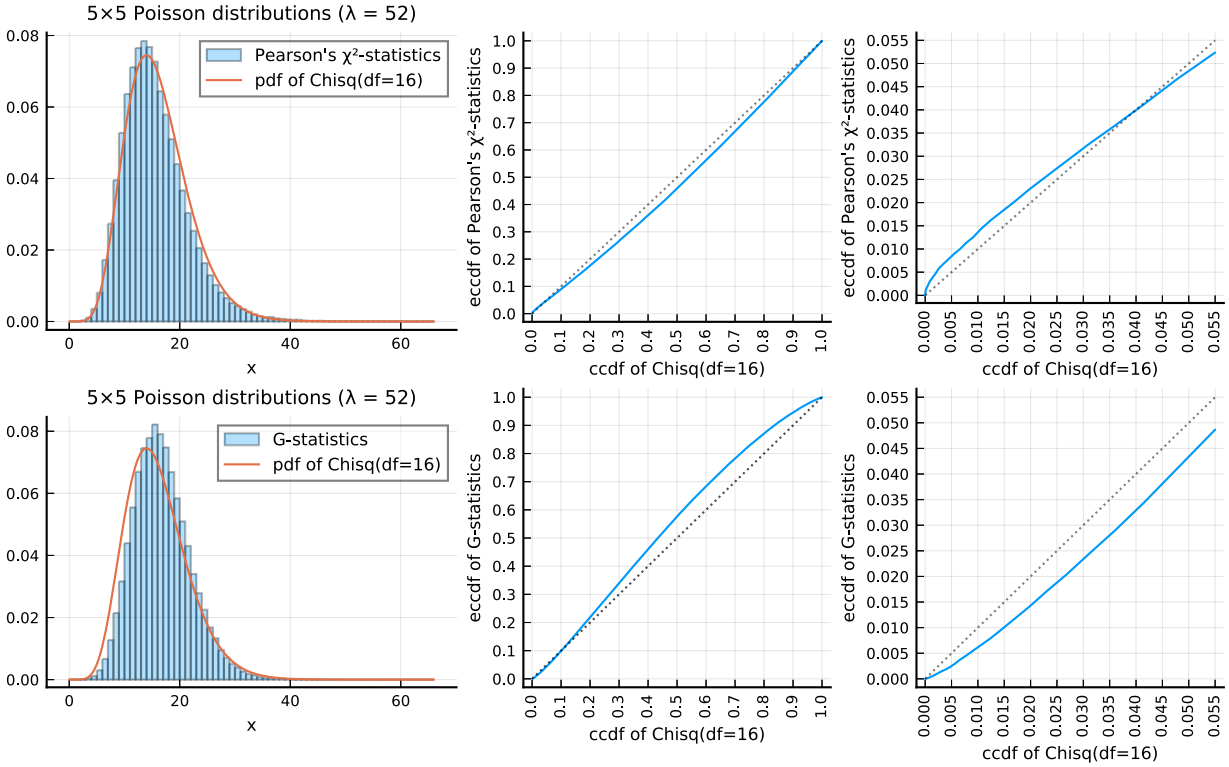
2.4.2 オリジナルの2倍の n=52 の場合

In [29]:

```
1  plot_sim_Poisson(λ=2n, P=P; binstep=1)
```

```
expectation = λ * P = [3.076923076923077 1.8461538461538463 1.5384615384615385 0.6153846153846154 0.9230769230769231;
10.76923076923077 6.461538461538462 5.384615384615385 2.153846153846154 3.230769230769231; 3.8461538461538463 2.30769
2307692308 1.9230769230769231 0.7692307692307693 1.153846153846154; 1.5384615384615385 0.9230769230769231 0.769230769
2307693 0.3076923076923077 0.46153846153846156; 0.7692307692307693 0.46153846153846156 0.38461538461538464 0.15384615
384615385 0.23076923076923078]
total = sum(expectation) = 52.0
(r, c) = size(expectation) = (5, 5)
df = df_chisq(expectation) = 16
  2.063475 seconds (3.10 M allocations: 166.322 MiB, 2.47% gc time)
```

Out[29]:

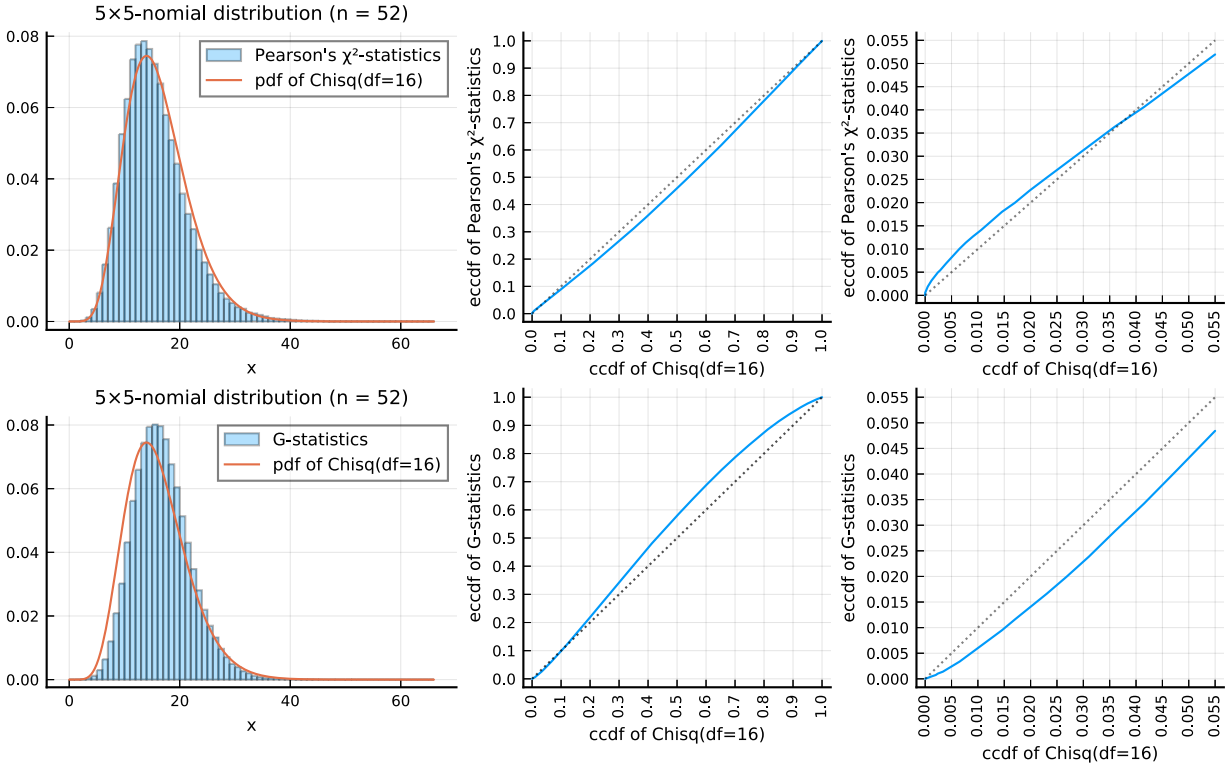


/

In [30]:
```
1  plot_sim_Multinomial(n=2n, P=P; binstep=1)
```

```
expectation = n * P = [3.076923076923077 1.8461538461538463 1.5384615384615385 0.6153846153846154 0.9230769230769231;
10.76923076923077 6.461538461538462 5.384615384615385 2.153846153846154 3.230769230769231; 3.8461538461538463 2.30769
2307692308 1.9230769230769231 0.7692307692307693 1.153846153846154; 1.5384615384615385 0.9230769230769231 0.769230769
2307693 0.3076923076923077 0.46153846153846156; 0.7692307692307693 0.46153846153846156 0.38461538461538464 0.15384615
384615385 0.23076923076923078]
total = sum(expectation) = 52.0
(r, c) = size(expectation) = (5, 5)
df = df_chisq(expectation) = 16
  0.787292 seconds (3.10 M allocations: 166.321 MiB, 5.53% gc time)
```

Out[30]:



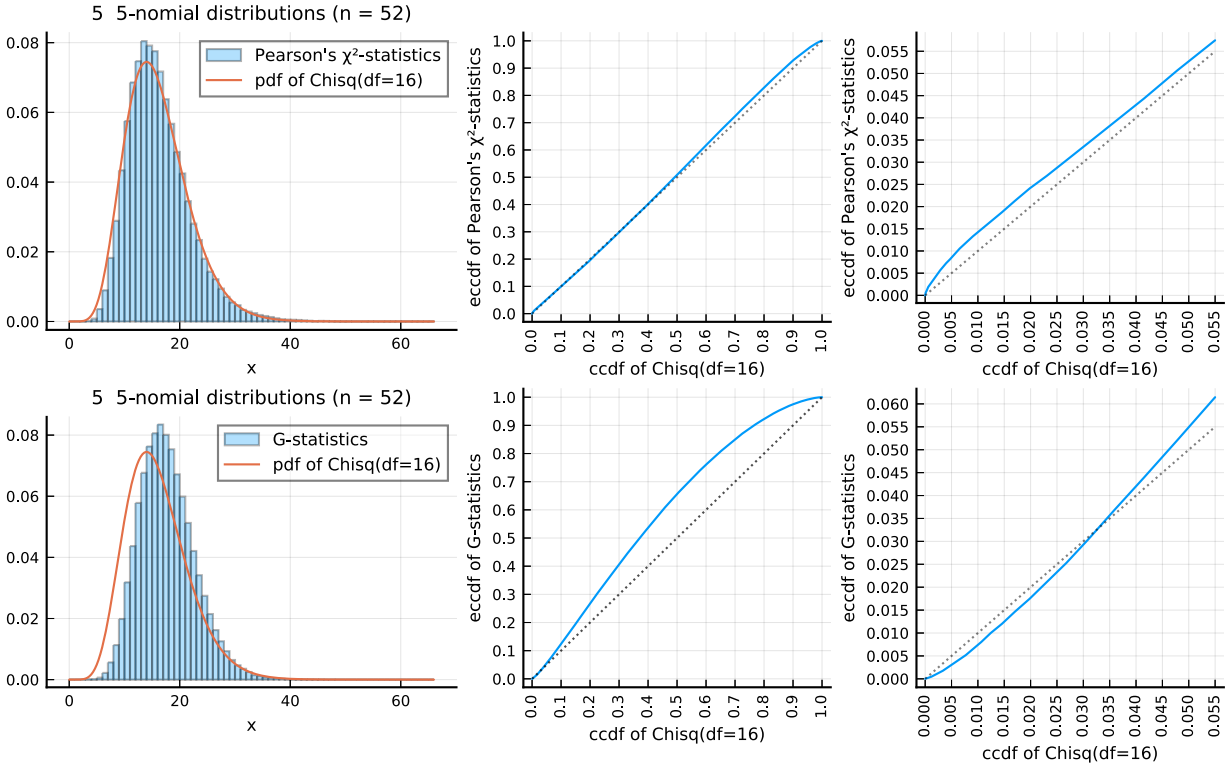
/

In [31]:

```
1  plot_sim_prod_Multinomial(M=2M, q=q; binstep=1)
```

```
expectation = M * q' = [3.076923076923077 1.8461538461538463 1.5384615384615385 0.6153846153846154 0.923076923076923
1; 10.76923076923077 6.461538461538462 5.384615384615385 2.153846153846154 3.230769230769231; 3.8461538461538463 2.30
7692307692308 1.9230769230769231 0.7692307692307693 1.153846153846154; 1.5384615384615385 0.9230769230769231 0.769230
7692307693 0.3076923076923077 0.46153846153846156; 0.7692307692307693 0.46153846153846156 0.38461538461538464 0.15384
615384615385 0.23076923076923078]
total = sum(expectation) = 52.0
(r, c) = size(expectation) = (5, 5)
df = df_chisq(expectation) = 16
  0.861111 seconds (3.40 M allocations: 218.201 MiB, 9.77% gc time)
```

Out[31]:

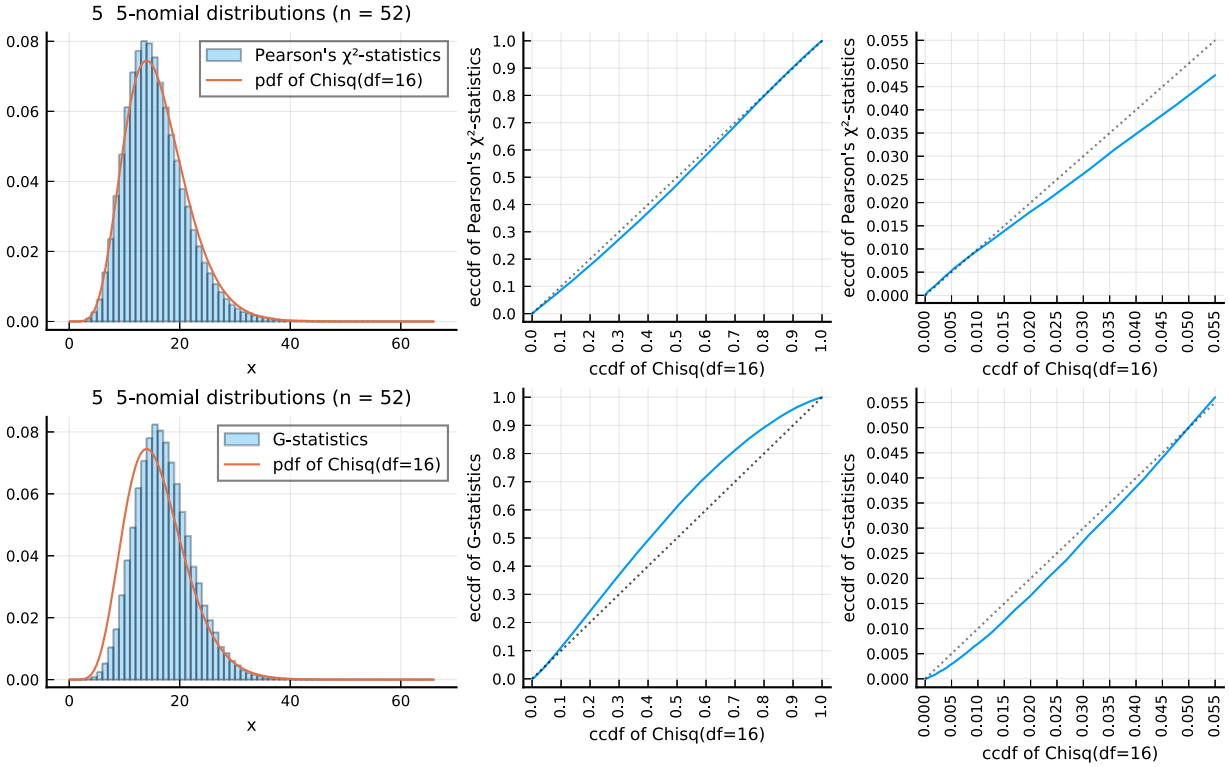


/

In [32]:
```
1  plot_sim_prod_Multinomial(M=2N, q=p; binstep=1)
```

```
expectation = M * q' = [3.076923076923077 10.769230769230768 3.8461538461538463 1.5384615384615385 0.7692307692307693; 1.8461538461538463 6.461538461538462 2.307692307692308 0.9230769230769231 0.46153846153846156; 1.5384615384615385 5.384615384615384 1.9230769230769231 0.7692307692307693 0.38461538461538464; 0.6153846153846154 2.1538461538461537 0.7692307692307693 0.3076923076923077 0.15384615384615385; 0.9230769230769231 3.230769230769231 1.153846153846154 0.46153846153846156 0.23076923076923078]
total = sum(expectation) = 52.0
(r, c) = size(expectation) = (5, 5)
df = df_chisq(expectation) = 16
  0.840041 seconds (3.40 M allocations: 218.201 MiB, 10.17% gc time)
```

Out[32]:



2.5 2×2 の場合の数値的確認

より詳細な比較については以下を参照せよ:

- 複数の確率分布でカイ二乗検定とG検定とFisherの正確検定を比較 2017-09-19〜20, 2019-10-10 (https://nbviewer.jupyter.org/gist/genkuroki/6924d68e12c87f3bbc10745ff0a183a6): カイ二乗検定とG検定とFisher検定の詳細な比較. 2019-10-10頃にmid-p版のFisher検定を追加.
- 2x2の分割表での独立性検定の比較 2017-09-26, 2019-10-14 (https://nbviewer.jupyter.org/gist/genkuroki/67f03274960dca00e73d5498ead138b7): カイ二乗検定とG検定とFisher検定の詳細な比較, (補正無しの)カイ二乗検定がかなりrobustであることがわかる. 2019-10-14にmid-p版のFisher検定を追加.
- 2×2の分割表における尤度函数 2017-09-26 (https://nbviewer.jupyter.org/gist/genkuroki/a3034d25a429b590d96c486064e53c8b): 尤度函数のプロット
- 2×2の分割表の独立性に関する様々な検定法の比較 2017-11-02 (https://nbviewer.jupyter.org/gist/genkuroki/3935a24dcfcb0fa4da46a0a3955158d8): カイ二乗検定とG検定とFisher検定以外に, Barnard検定とBoschloo検定も比較してみた. 単純なカイ二乗検定で十分だと思われる.

In [33]:
```
1  @show M = [10, 15]
2  @show q = [0.1, 0.9]
3  P = param_indep(M/sum(M), q)
```

```
M = [10, 15] = [10, 15]
q = [0.1, 0.9] = [0.1, 0.9]
```

Out[33]:
```
2×2 Array{Float64,2}:
 0.04  0.36
 0.06  0.54
```

2.5.1 $n = 50$ の場合

/

In [34]:

```
E = round.(Int, 2M*q')
display("text/html", raw"$\text{期待値} = " * sympy.latex(Sym.(E)) * raw"$")
```

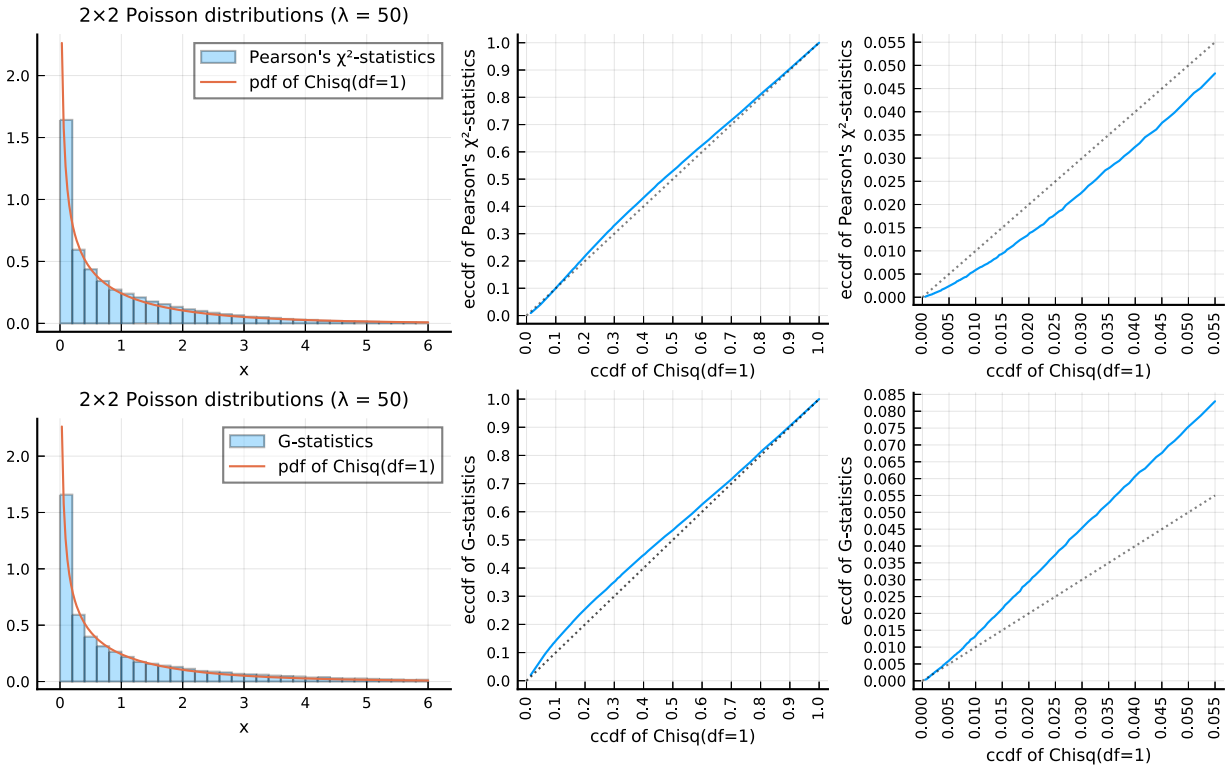
期待値 $= \begin{bmatrix} 2 & 18 \\ 3 & 27 \end{bmatrix}$

In [35]:

```
plot_sim_Poisson(λ=50, P=P, binstep=0.2)
```

```
expectation = λ * P = [2.0000000000000004 18.000000000000004; 3.0 27.0]
total = sum(expectation) = 50.0
(r, c) = size(expectation) = (2, 2)
df = df_chisq(expectation) = 1
  0.563940 seconds (3.10 M allocations: 137.330 MiB, 6.30% gc time)
```

Out[35]:

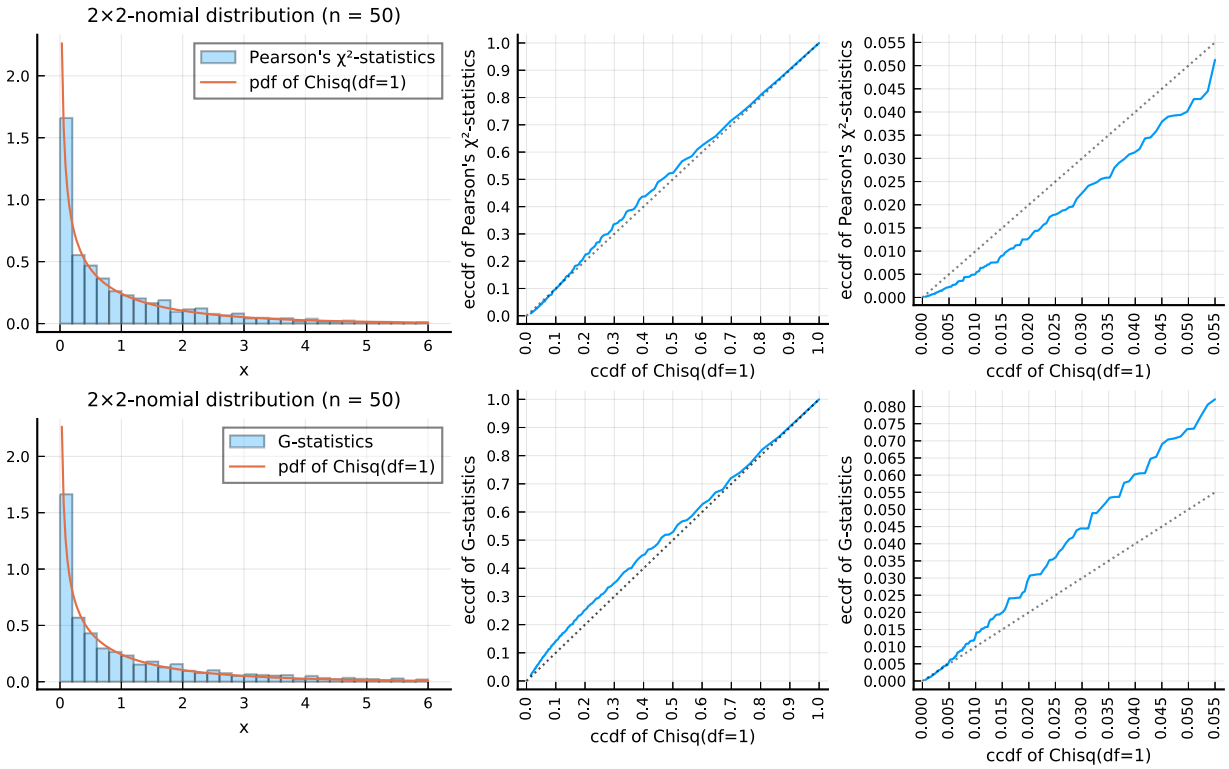


/

In [36]:
```
1  plot_sim_Multinomial(n=50, P=P, binstep=0.2)
```

```
expectation = n * P = [2.0000000000000004 18.000000000000004; 3.0 27.0]
total = sum(expectation) = 50.0
(r, c) = size(expectation) = (2, 2)
df = df_chisq(expectation) = 1
  0.366426 seconds (3.10 M allocations: 137.329 MiB, 12.20% gc time)
```

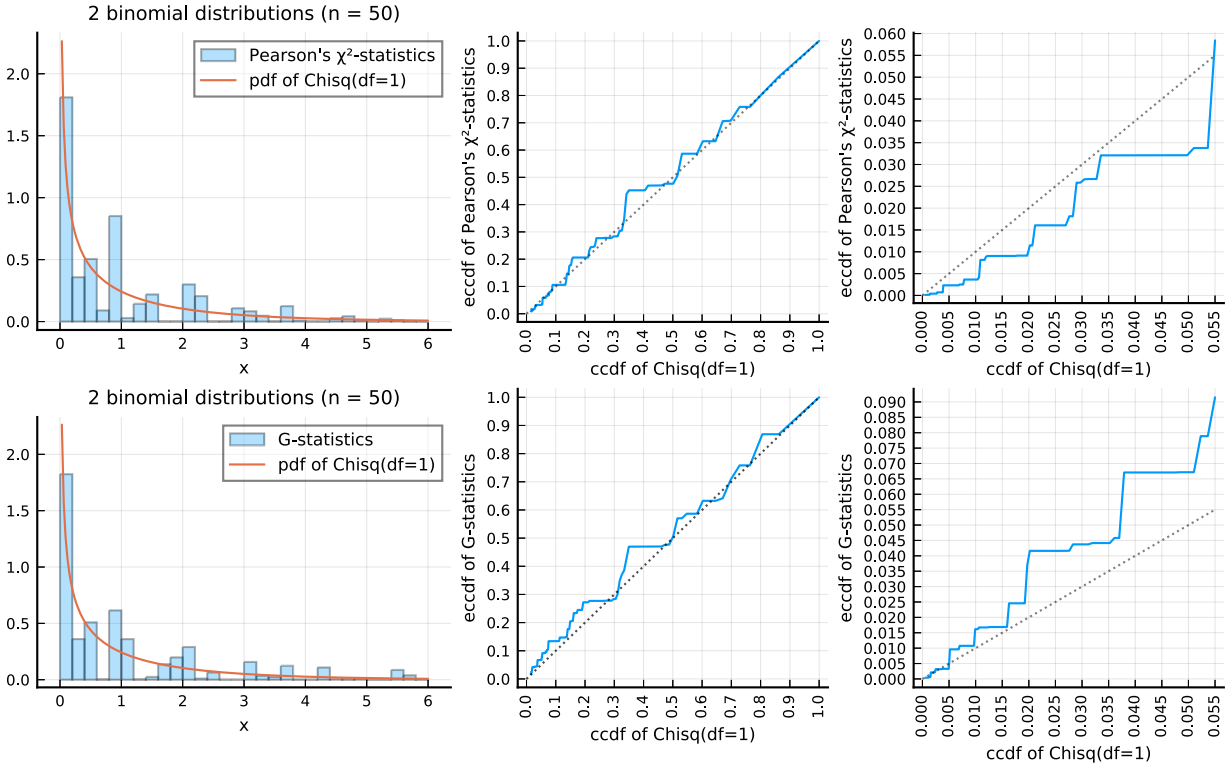
Out[36]:



In [37]:
```
1  plot_sim_prod_Multinomial(M=2M, q=q, binstep=0.2)
```

```
expectation = M * q' = [2.0 18.0; 3.0 27.0]
total = sum(expectation) = 50.0
(r, c) = size(expectation) = (2, 2)
df = df_chisq(expectation) = 1
  0.353167 seconds (3.10 M allocations: 146.485 MiB, 11.12% gc time)
```

Out[37]:

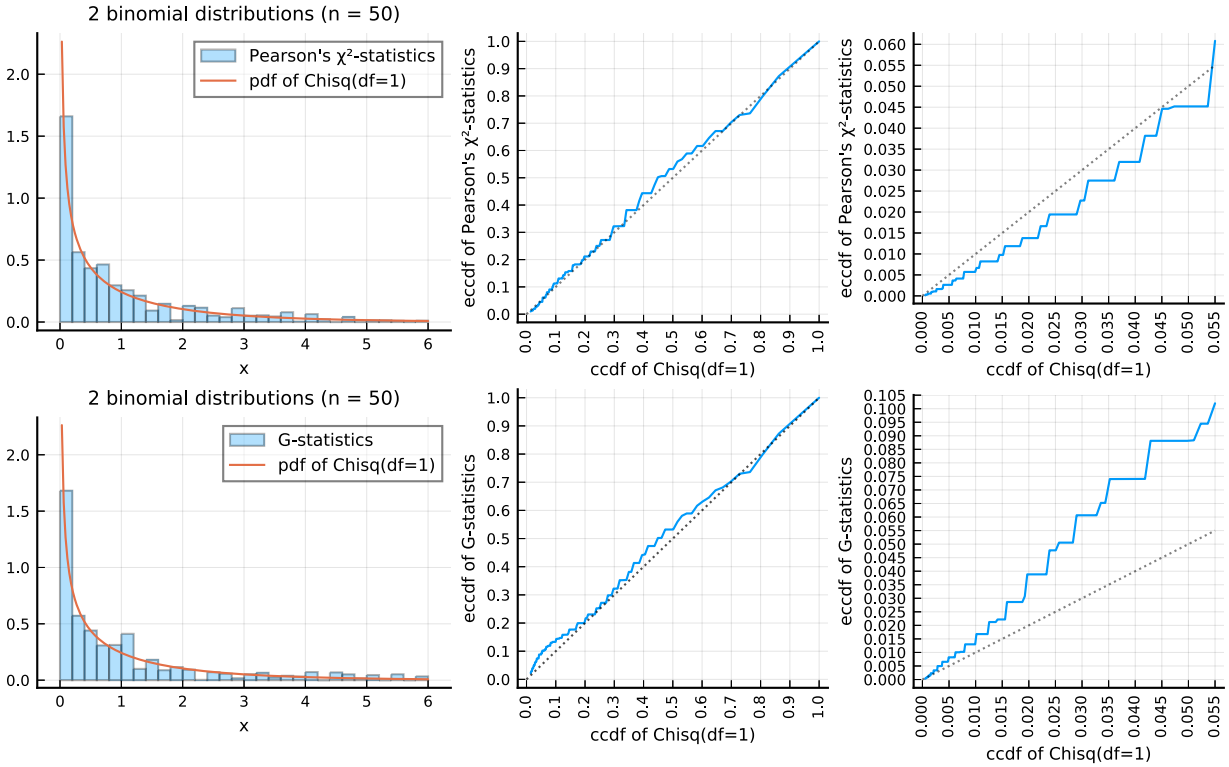


/

In [38]:
```
1  plot_sim_prod_Multinomial(M=round.(Int, sum(2M)*q), q=M/sum(M), binstep=0.2)
```

```
expectation = M * q' = [2.0 3.0; 18.0 27.0]
total = sum(expectation) = 50.0
(r, c) = size(expectation) = (2, 2)
df = df_chisq(expectation) = 1
  0.365518 seconds (3.10 M allocations: 146.485 MiB, 11.57% gc time)
```

Out[38]:



2.5.2 $n = 100$ の場合

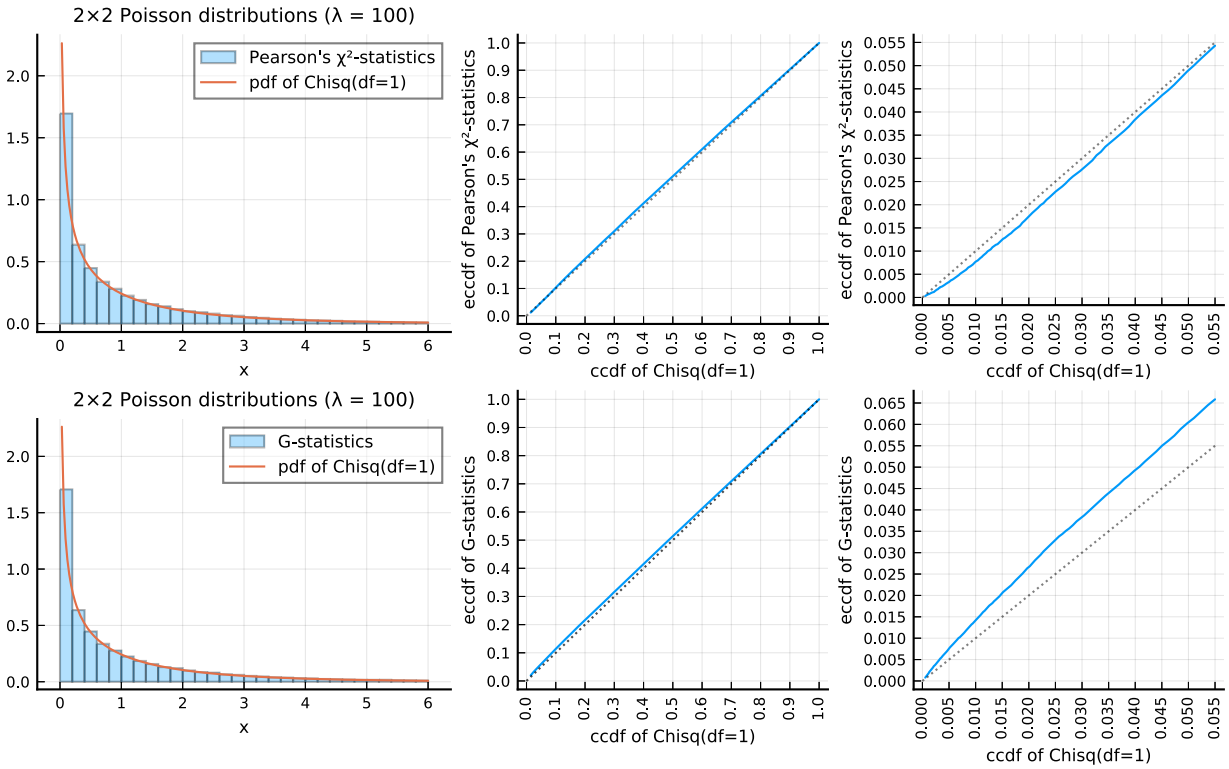
In [39]:
```
1  E = round.(Int, 4M*q')
2  display("text/html", raw"$\text{期待値} = " * sympy.latex(Sym.(E)) * raw"$")
```

期待値 $= \begin{bmatrix} 4 & 36 \\ 6 & 54 \end{bmatrix}$

In [40]:
```
1  plot_sim_Poisson(λ=100, P=P, binstep=0.2)
```

```
expectation = λ * P = [4.000000000000001 36.00000000000001; 6.0 54.0]
total = sum(expectation) = 100.0
(r, c) = size(expectation) = (2, 2)
df = df_chisq(expectation) = 1
  0.684127 seconds (3.10 M allocations: 137.330 MiB, 13.31% gc time)
```

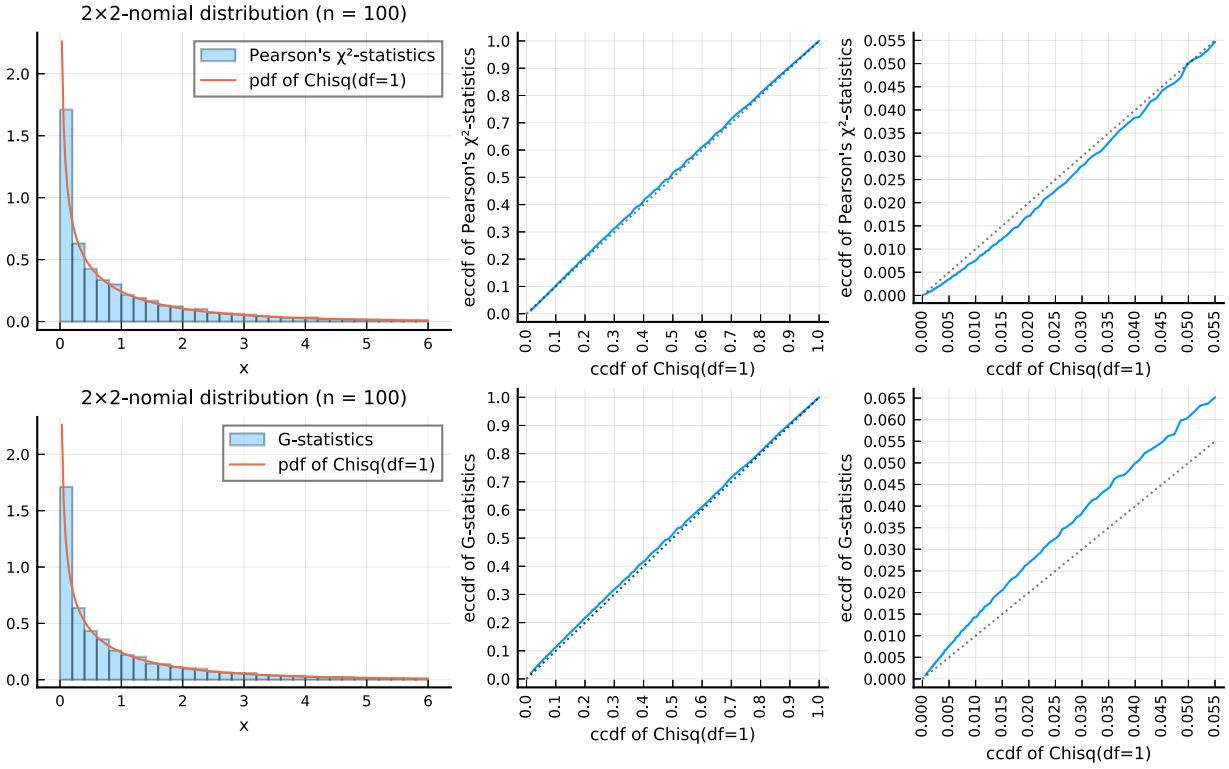
Out[40]:



In [41]:
```
1  plot_sim_Multinomial(n=100, P=P, binstep=0.2)
```

```
expectation = n * P = [4.000000000000001 36.00000000000001; 6.0 54.0]
total = sum(expectation) = 100.0
(r, c) = size(expectation) = (2, 2)
df = df_chisq(expectation) = 1
  0.352802 seconds (3.10 M allocations: 137.329 MiB, 11.88% gc time)
```

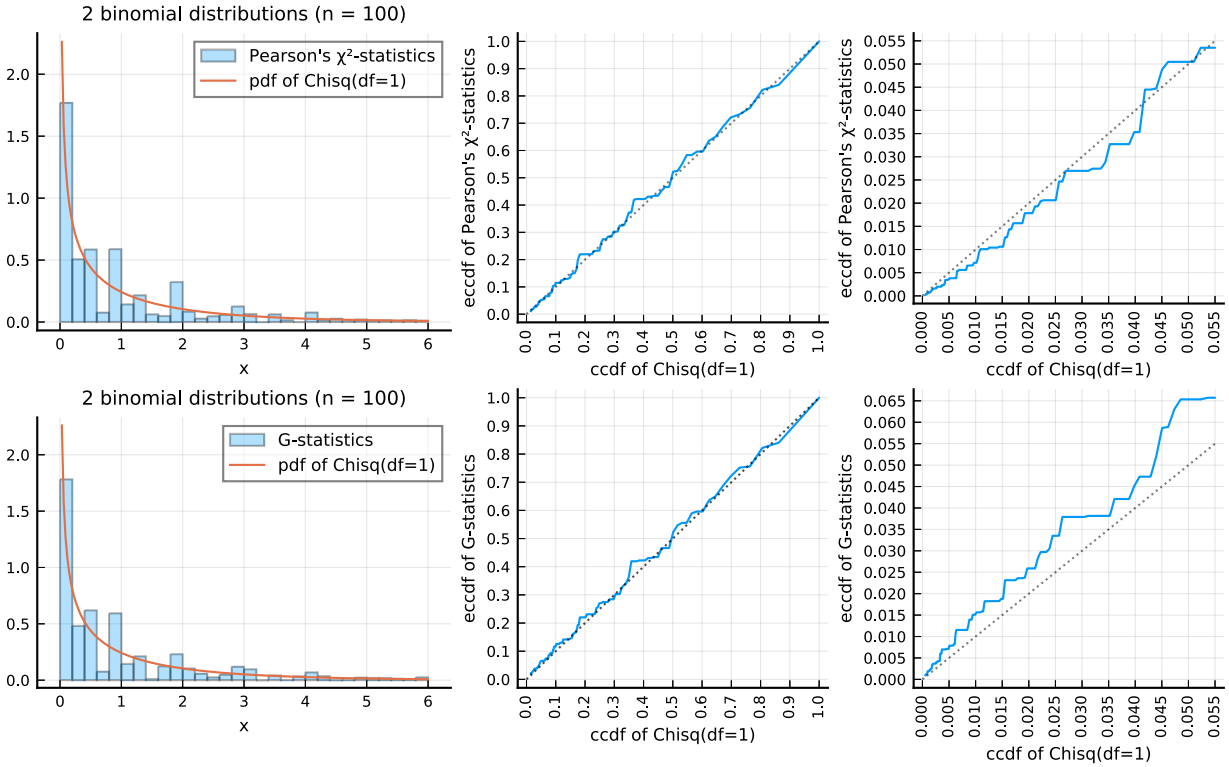
Out[41]:



In [42]:
```
1  plot_sim_prod_Multinomial(M=4M, q=q, binstep=0.2)
```

```
expectation = M * q' = [4.0 36.0; 6.0 54.0]
total = sum(expectation) = 100.0
(r, c) = size(expectation) = (2, 2)
df = df_chisq(expectation) = 1
  0.357124 seconds (3.10 M allocations: 146.485 MiB, 10.65% gc time)
```

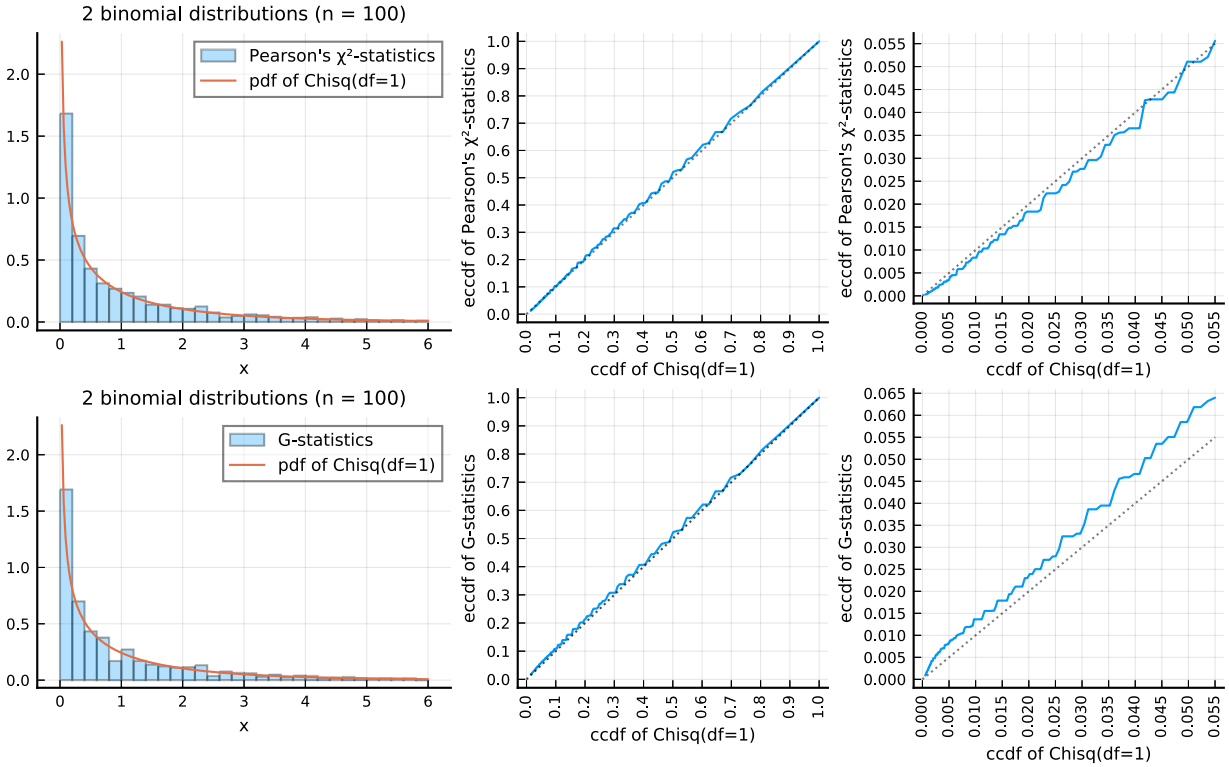
Out[42]:



In [43]:
```
1  plot_sim_prod_Multinomial(M=round.(Int, sum(4M)*q), q=M/sum(M), binstep=0.2)
```

```
expectation = M * q' = [4.0 6.0; 36.0 54.0]
total = sum(expectation) = 100.0
(r, c) = size(expectation) = (2, 2)
df = df_chisq(expectation) = 1
  0.404579 seconds (3.10 M allocations: 146.485 MiB, 9.92% gc time)
```

Out[43]:



/

3 分割表における対数尤度比の計算

3.1 分割表 $A = [a_{ij}]$ に制限がない場合

rc 個のPoisson分布の直積:

$$p(A|\Lambda) = \prod_{i,j} \frac{e^{-\lambda_{ij}} \lambda_{ij}^{a_{ij}}}{a_{ij}!},$$
$$\log p(A|\Lambda) = \sum_{i,j} (a_{ij} \log \lambda_{ij} - \lambda_{ij} - \log a_{ij}!).$$

パラメーター: $\Lambda = [\lambda_{ij}]$, λ_{ij} は非負の実数.

独立性を満たすパラメーター: 正の実数 λ と非負の実数 μ_i, ν_j たちで $\sum_i \mu_i = \sum_j \nu_j = \lambda$ を満たすもの. 上の λ_{ij} との関係は

$$\lambda_{ij} = \frac{\mu_i \nu_j}{\lambda}.$$

このとき

$$\lambda = \sum_{i,j} \lambda_{ij}, \quad \mu_i = \sum_j \lambda_{ij}, \quad \nu_i = \sum_i \lambda_{ij}.$$

3.1.1 rc 個のPoisson分布の直積モデルでの最尤法の解

対数尤度

$$L = \log p(A|\Lambda) = \sum_{i,j} (a_{ij} \log \lambda_{ij} - \lambda_{ij} - \log a_{ij}!)$$

を最大にするパラメーター $\lambda_{ij} = \hat{\lambda}_{ij}$ を求めよう.

$$\frac{\partial L}{\partial \lambda_{ij}} = \frac{a_{ij}}{\lambda_{ij}} - 1$$

より,

$$\hat{\lambda}_{ij} = a_{ij}.$$

3.1.2 独立性を満たすパラメーターに制限された rc 個のPoisson分布の直積モデルでの最尤法の解

独立性 $\lambda_{ij} = \mu_i \nu_j / \lambda, \lambda = \sum_{i,j} \lambda_{ij}$ を満たすパラメーターに制限した場合の対数尤度を最大にするパラメーター $\lambda = \tilde{\lambda}, \mu_i = \tilde{\mu}_i, \nu_j = \tilde{\nu}_j, \tilde{\lambda}_{ij} = \tilde{\mu}_i \tilde{\nu}_j / \tilde{\lambda}$ をLagrangeの未定乗数法で求めよう. そのために

$$M = \sum_{i,j} \left(a_{ij} (\log \mu_i + \log \nu_j - \log \lambda) - \frac{\mu_i \nu_j}{\lambda} \right) - \alpha \left(\sum_i \mu_i - \lambda \right) - \beta \left(\sum_j \nu_j - \lambda \right)$$

とおくと,

$$\frac{\partial M}{\partial \alpha} = -\sum_i \mu_i + \lambda, \quad \frac{\partial M}{\partial \beta} = -\sum_j \nu_j + \lambda,$$
$$\frac{\partial M}{\partial \lambda} = 1 - \frac{\sum_{i,j} a_{ij}}{\lambda} + \alpha + \beta,$$
$$\frac{\partial M}{\partial \mu_i} = \frac{\sum_j a_{ij}}{\mu_i} - 1 - \alpha, \quad \frac{\partial M}{\partial \nu_j} = \frac{\sum_i a_{ij}}{\nu_j} - 1 - \beta$$

なので,

$$\tilde{\lambda} = \sum_{i,j} a_{ij}, \quad \tilde{\mu}_i = \sum_j a_{ij}, \quad \tilde{\nu}_j = \sum_j a_{ij}, \quad \tilde{\lambda}_{ij} = \frac{\tilde{\mu}_i \tilde{\nu}_j}{\tilde{\lambda}}.$$

この場合には $\alpha = \beta = 0$ となる.

3.1.3 rc 個のPoisson分布の直積モデルの場合の対数尤度比

rc 個のPoisson分布の直積モデルの場合の対数尤度比は $\hat{\Lambda} = [\hat{\lambda}_{ij}] = [a_{ij}] = A$, $\tilde{\Lambda} = [\tilde{\lambda}_{ij}]$ とおくとき, 以下のようになる:

$$G = 2(\log p(A|\hat{\Lambda}) - \log p(A|\tilde{\Lambda})) = 2\sum_{i,j} a_{ij} \log \frac{a_{ij}}{\tilde{\lambda}_{ij}}.$$

ここで $\sum_{i,j} a_{ij} = \sum_{i,j} \tilde{\lambda}_{ij}$ を使った.

分割表 $A = [a_{ij}]$ が独立性を満たすパラメーターに対する rc 個のPoisson分布の直積に従う確率変数であるとき, Wilksの定理より(もしくは直接的な計算によって), G およびそれを近似する

$$X^2 = \sum_{i,j} \frac{(a_{ij} - \tilde{\lambda}_{ij})^2}{\tilde{\lambda}_{ij}}$$

はサンプルサイズを大きくするとき漸近的に自由度 $(r-1)(c-1)$ のχ²分布に従う.

3.2 分割表 $A = [a_{ij}]$ の総和 $\sum_{i,j} a_{ij} = n$ が一定の場合

分割表の制限: $\sum_{i,j} a_{ij} = n$ が一定.

rc 項分布:

$$p(A|n,P) = \frac{n!}{n^n} \prod_{i,j} \frac{\lambda_{ij}^{a_{ij}}}{a_{ij}!},$$
$$\log p(A|n,P) = \sum_{i,j} (a_{ij} \log \lambda_{ij} - \log a_{ij}!) + \log n! - n \log n.$$

パラメーター: 非負の実数 λ_{ij} で $\sum_{i,j} \lambda_{ij} = n$ を満たすもの. ただし $P = [p_{ij}]$ と λ_{ij} の関係は $\lambda_{ij} = np_{ij}$.

独立性を満たすパラメーター: 非負の実数 μ_i, ν_j 達で $\sum_i \mu_i = \sum_j \nu_j = n$ を満たすもの. ただし λ_{ij} との関係は

$$\lambda_{ij} = \frac{\mu_i \nu_j}{n}.$$

3.2.1 rc 項分布モデルの最尤法の解

対数尤度を最大にする総和が n のパラメーター $\lambda_{ij} = \hat{\lambda}_{ij}$ を求めるためにLagrangeの未定乗数法を使う. そのために

$$L = \sum_{i,j} a_{ij} \log \lambda_{ij} - \alpha \left(\sum_{i,j} \lambda_{ij} - n \right)$$

とおくと,

$$\frac{\partial L}{\partial \alpha} = -\sum_{i,j} \lambda_{ij} + n, \quad \frac{\partial L}{\partial \lambda_{ij}} = \frac{a_{ij}}{\lambda_{ij}} - \alpha$$

なので,

$$\hat{\lambda}_{ij} = a_{ij}.$$

3.2.2 独立性を満たすパラメーターに制限された rc 項分布モデルの最尤法の解

独立性 $\lambda_{ij} = \mu_i \nu_j / n$, $\sum_{i,j} \lambda_{ij} = \sum_{i,j} a_{ij} = n$ を満たすパラメーターに制限した場合の対数尤度を最大にするもの $\mu_i = \tilde{\mu}_i$, $\nu_j = \tilde{\nu}_j$, $\tilde{\lambda}_{ij} = \tilde{\mu}_i \tilde{\nu}_j / n$ をLagrangeの未定乗数法で求めよう. そのために

$$M = \sum_{i,j} a_{ij} (\log \mu_i + \log \nu_j) - \alpha \left(\sum_i \mu_i - n \right) - \beta \left(\sum_i \nu_i - n \right)$$

とおくと,

$$\frac{\partial M}{\partial \alpha} = -\sum_i \mu_i + n, \quad \frac{\partial M}{\partial \beta} = -\sum_j \nu_i + n,$$
$$\frac{\partial M}{\partial \mu_i} = \frac{\sum_j a_{ij}}{\mu_i} - \alpha, \quad \frac{\partial M}{\partial \nu_j} = \frac{\sum_i a_{ij}}{\nu_j} - \beta.$$

なので,

$$\tilde{\mu}_i = \sum_j a_{ij}, \quad \tilde{\nu}_j = \sum_i a_{ij}, \quad \tilde{\lambda}_{ij} = \frac{\tilde{\mu}_i \tilde{\nu}_j}{n}.$$

この場合には $\alpha = \beta = 1$ になる.

3.2.3 rc 項分布モデルの場合の対数尤度比

rc 項分布モデルの場合の対数尤度比は $\hat{\Lambda} = [\hat{\lambda}_{ij}] = [a_{ij}] = A$, $\tilde{\Lambda} = [\tilde{\lambda}_{ij}]$, $\hat{P} = \hat{\Lambda}/n$, $\tilde{P} = \tilde{\Lambda}/n$ とおくとき, 以下のようになる:

$$G = 2(\log p(A|n, \hat{P}) - \log p(A|n, \tilde{P})) = 2 \sum_{i,j} a_{ij} \log \frac{a_{ij}}{\tilde{\lambda}_{ij}}.$$

分割表 $A = [a_{ij}]$ が独立性を満たすパラメーターに対する rc 項分布に従う確率変数であるとき, Wilksの定理より(もしくは直接的な計算によって), G およびそれを近似する

$$X^2 = \sum_{i,j} \frac{(a_{ij} - \tilde{\lambda}_{ij})^2}{\tilde{\lambda}_{ij}}$$

はサンプルサイズを大きくするとき漸近的に自由度 $(r-1)(c-1)$ のχ²分布に従う.

3.3 分割表 $A = [a_{ij}]$ の行の和 $\sum_j a_{ij} = \mu_i$ がすべて一定の場合

分割表の制限: $\sum_j a_{ij} = \mu_i$ がすべて一定. $n = \sum_i \mu_i$, $\mu = (\mu_1, \ldots, \mu_r)$ とおく.

r 個の c 項分布の直積:

$$p(A|\mu, Q) = \prod_i \frac{\mu_i!}{\mu_i^{\mu_i}} \cdot \prod_{i,j} \frac{\lambda_{ij}^{a_{ij}}}{a_{ij}!},$$
$$\log p(A|\mu, Q) = \sum_{i,j} (a_{ij} \log \lambda_{ij} - \log a_{ij}!) + \sum_i (\log \mu_i! - \mu_i \log \mu_i).$$

パラメーター: 非負の実数 λ_{ij} 達で $\sum_j \lambda_{ij} = \mu_i$ を満たすもの. ただし, $Q = [q_{ij}]$ と λ_{ij} の関係は $\lambda_{ij} = \mu_i q_{ij}$.

独立性を満たすパラメーター: 非負の実数 ν_j 達で $\sum_j \nu_j = n$ を満たすもの. ただし λ_{ij} との関係は

$$\lambda_{ij} = \frac{\mu_i \nu_j}{n}.$$

3.3.1 r 個の c 項分布モデルの最尤法の解

$\sum_j \lambda_{ij} = \mu_i$ がすべて一定という条件を満たすパラメーター λ_{ij} で対数尤度を最大にするもの $\lambda_{ij} = \tilde{\lambda}_{ij}$ を求めるためにLagrangeの未定乗数法を使おう. そのために

$$L = \sum_{i,j} a_{ij} \log \lambda_{ij} - \sum_i \alpha_i \left(\sum_j \lambda_{ij} - \mu_i \right)$$

とおくと,

$$\frac{\partial L}{\partial \alpha_i} = -\sum_j \lambda_{ij} + \mu_i, \quad \frac{\partial L}{\partial \lambda_{ij}} = \frac{a_{ij}}{\lambda_{ij}} - \alpha_i$$

なので

$$\tilde{\lambda} = a_{ij}.$$

この場合には $\sum_j a_{ij} = \mu_i$ より $\alpha_i = 1$ となる.

3.3.2 独立性を満たすパラメーターに制限された r 個の c 項分布モデルの最尤法の解

独立性 $\lambda_{ij} = \mu_i \nu_j / n$, $\sum_j \lambda_{ij} = \sum_j a_{ij} = \mu_i$ を満たすパラメーターに制限した場合の対数尤度を最大にするもの $\nu_j = \tilde{\nu}_j$, $\tilde{\lambda}_{ij} = \mu_i \tilde{\nu}_j / n$ をLagrangeの未定乗数法で求めよう. そのために

$$M = \sum_{i,j} a_{ij} \log \nu_j - \alpha \left(\sum_j \nu_j - n \right)$$

とおくと

$$\frac{\partial L}{\partial \alpha} = -\sum_j \nu_j + n, \quad \frac{\partial L}{\partial \nu_j} = \frac{\sum_i a_{ij}}{\nu_j} - \alpha$$

なので,

$$\tilde{\nu}_j = \sum_i a_{ij}, \quad \tilde{\lambda}_{ij} = \frac{\mu_i \tilde{\nu}_j}{n}.$$

3.3.3 r 個の c 項分布の直積モデルの場合の対数尤度比

r 個の c 項分布の直積モデルの場合の対数尤度比は $\hat{\Lambda} = [\hat{\lambda}_{ij}] = [a_{ij}] = A$, $\tilde{\Lambda} = [\tilde{\lambda}_{ij}]$, $\hat{Q} = [\hat{q}_{ij}]$, $\hat{q}_{ij} = \hat{\lambda}_{ij}/\mu_i$, $\tilde{Q} = [\tilde{q}_{ij}]$, $\tilde{q}_{ij} = \tilde{\lambda}_{ij}/\mu_i = \tilde{\nu}_j/n$ とおくとき, 以下のようになる:

$$G = 2(\log p(A|\mu, \hat{Q}) - \log p(A|\mu, \tilde{Q})) = 2 \sum_{i,j} a_{ij} \log \frac{a_{ij}}{\tilde{\lambda}_{ij}}.$$

分割表 $A = [a_{ij}]$ が独立性を満たすパラメーターに対する r 個の c 項分布の直積に従う確率変数であるとき, Wilksの定理より(もしくは直接的な計算によって), この G およびそれを近似する

$$X^2 = \sum_{i,j} \frac{(a_{ij} - \tilde{\lambda}_{ij})^2}{\tilde{\lambda}_{ij}}$$

はサンプルサイズを大きくするとき漸近的に自由度 $(r-1)(c-1)$ のχ²分布に従う.

3.4 分割表 $A = [a_{ij}]$ の周辺度数がすべて固定されている場合

分割表の制限: $\sum_j a_{ij} = \mu_i$ と $\sum_i a_{ij} = \nu_j$ がすべて一定. $n = \sum_i \mu_i = \sum_j \nu_j$, $\mu = (\mu_1, \dots, \mu_r)$, $\nu = (\nu_1, \dots, \nu_c)$ とおく.

周辺度数がすべて固定されている場合の分割表の確率分布:

$$p(A|\mu, \nu, \Lambda) = \frac{1}{Z(\Lambda)} \prod_{i,j} \frac{\lambda_{ij}^{a_{ij}}}{a_{ij}!}, \quad Z(\Lambda) = \sum_A \prod_{i,j} \frac{\lambda_{ij}^{a_{ij}}}{a_{ij}!}.$$

パラメーター: 正の実数 λ_{ij} で $\sum_j \lambda_{ij} = \mu_i$, $\sum_i \lambda_{ij} = \nu_j$ を満たすもの.

独立性を満たすパラメーター: この場合には独立性を満たすパラメーターは μ_i, ν_j から次のように一意に決まってしまう:

$$\lambda_{ij} = \frac{\mu_i \nu_j}{n}.$$

このとき, 分割表の確率分布は次の形になる:

$$p(A|\mu, \nu, \Lambda) = p(A|\mu, \nu) = \frac{\prod_i \mu_i! \cdot \prod_j \nu_j!}{n! \prod_{i,j} a_{ij}!} = \frac{\prod_{j=1}^{c} \binom{\nu_j}{a_{1j}, \dots, a_{rj}}}{\binom{n}{\mu_1, \dots, \mu_r}}.$$

分割表 $A = [a_{ij}]$ がこの独立性を満たすパラメーター $\lambda_{ij} = \mu_i \nu_j / n$ に対する周辺度数 μ_i, ν_j 達がすべて固定されている場合の分割表の確率分布に従う確率変数であるとき,

$$G = 2 \sum_{i,j} a_{ij} \log \frac{a_{ij}}{\lambda_{ij}}, \quad X^2 = \sum_{i,j} \frac{(a_{ij} - \lambda_{ij})^2}{\lambda_{ij}}$$

がサンプルサイズを大きくするとき漸近的に自由度 $(r-1)(c-1)$ のχ²分布に従うはすでに上の方で示してあった.