# 生起回数が負の二項分布に従う事象のシミュレーション

Hirotsugu Seike

2026 年 1 月 1 日

## 1　導入

負の二項分布は, 偶然が起きる「環境」のバラつきを考慮した事象の発生回数が従う分布である. 例えば, 1 日の間にパソコン上で発生するエラー数, 日毎の SNS 投稿数・感染数・事故の件数などのモデル化に用いられる. ここでは, 事象の生起回数 $N$ を確率変数として考え, $N$ がパラメータ $\alpha, p$ $(\alpha > 0, 0 < p < 1)$ の負の二項分布に従うことを $N \sim \mathrm{NB}(\alpha, p)$ で表す.

上記を表現するため, ある一定期間内での事象の平均生起回数 $\Lambda$ が形状母数 $\alpha$, 尺度母数 $1/\beta(>0)$ のガンマ分布に従うと仮定し, 事象の生起パターンがポアソン分布と同様であると仮定する. 言い換えれば, 次式が成立する.

$$\mathrm{P}(N = n) = \int_0^\infty \frac{\lambda^n \exp(-\lambda)}{n!} \cdot \frac{\beta^\alpha}{\Gamma(\alpha)} \cdot \lambda^{\alpha-1} \cdot e^{-\beta\lambda} d\lambda, \tag{1}$$

$$= \frac{\beta^\alpha}{n!\Gamma(\alpha)} \int_0^\infty \lambda^{n+\alpha-1} \cdot e^{-(\beta+1)\lambda} d\lambda, \tag{2}$$

$$= \frac{\beta^\alpha}{n!\Gamma(\alpha)} \cdot \frac{\Gamma(n+\alpha)}{(\beta+1)^{n+\alpha}}. \tag{3}$$

ここで, $p = \beta/(\beta+1)$ と定義すれば, 次式が得られる.

$$\mathrm{P}(N = n) = \frac{\Gamma(n+\alpha)}{\Gamma(n+1)\Gamma(\alpha)} \cdot \left(\frac{\beta}{\beta+1}\right)^\alpha \cdot \left(\frac{1}{\beta+1}\right), \tag{4}$$

$$=_{n+\alpha-1} C_{\alpha-1} \cdot p^\alpha \cdot (1-p)^n. \tag{5}$$

## 2　シミュレーション

生起回数が負の二項分布に従う事象の離散時間シミュレーションは, 例えば図 1 のようにして実施できる. 固定のタイムフレーム $T$ 毎に, 平均生起回数 $\Lambda$ をガンマ分布に従う確率変数として抽出し, 時間 $T$ が経過するまで, 平均 $1/\Lambda$ の指数分布に従う事象を発生させている.

```
import numpy as np
from collections import Counter
import math

# ======================================================
# シミュレーション本体
# ======================================================

def simulate_one_trial(alpha, beta, T):
```

```
    """
␣␣␣␣試行分のイベント生成1
␣␣␣␣alpha,␣beta␣:␣Gamma(alpha,␣beta)␣の␣shape,␣rate
␣␣␣␣T␣␣␣␣␣␣␣:␣タイムフレーム
␣␣␣␣Lambda␣␣:␣T␣あたりの平均事象数
␣␣␣␣"""
    # Lambda ~ （あたりの平均事象数）GammaT
    Lambda = np.random.gamma(shape=alpha, scale=1.0 / beta)

    # 単位時間あたりの rate
    rate = Lambda / T

    t = 0.0
    event_times = []

    while True:
        # 平均 T / Lambda の指数分布
        t += np.random.exponential(scale=1.0 / rate)

        if t > T:
            break

        event_times.append(t)

    return Lambda, event_times


# =====================================================
# 負の二項分布（理論値）: PMF
# N(T) ~ NB(alpha, p),  p = beta / (beta + 1)
# =====================================================

def negbin_pmf(n, alpha, beta):
    p = beta / (beta + 1.0)
    return (
        math.gamma(n + alpha)
        / (math.gamma(alpha) * math.factorial(n))
        * (p ** alpha)
        * ((1 - p) ** n)
    )


# =====================================================
# 検証用コード
# =====================================================

if __name__ == "__main__":
    # パラメータ
    alpha = 3.0
    beta = 2.0
    T = 4.0

    n_trials = 20000

    counts = []
    lambdas = []

    for _ in range(n_trials):
```

```
        lam, events = simulate_one_trial(alpha, beta, T)
        lambdas.append(lam)
        counts.append(len(events))

    counts = np.array(counts)

    # ---- 基本統計量 ----
    print("=== Basic statistics ===")
    print("Mean count      :", counts.mean())
    print("Variance count  :", counts.var())
    print("Mean lambda     :", np.mean(lambdas))
    print()

    # 理論値
    theo_mean = alpha / beta
    theo_var = theo_mean + (theo_mean ** 2) / alpha

    print("=== Theoretical (Negative Binomial) ===")
    print("Theoretical mean :", theo_mean)
    print("Theoretical var  :", theo_var)
    print()

    # ---- 分布比較（最初の数点）----
    print("=== Empirical vs Theoretical PMF ===")
    freq = Counter(counts)
    max_k = 8

    for k in range(max_k + 1):
        emp = freq[k] / n_trials
        theo = negbin_pmf(k, alpha, beta)
        print(f"k={k:2d}   empirical={emp:.4f}   theoretical={theo:.4f}")
```

図 1: Simulation code for events whose counts follow a negative binomial distribution.

## 参考文献

## 付録 A　特になし