Practical Examples

Dr. G. Omprakash

Assistant Professor, ECE, KLEF

Table of Contents

Bank arrivals: Staffing decisions using Poisson

Bank arrivals


```
# Bank arrivals (Poisson) - sizing tellers
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import poisson

np.random.seed(1)
lam = 10                    # true average customers per hour
hours = 1000                # simulated hourly observations
arrivals = np.random.poisson(lam, size=hours)

# sample statistics
sample_mean = arrivals.mean()
sample_var = arrivals.var(ddof=0)

# capacity example
tellers = 2
service_rate = 4            # customers per hour per teller
capacity = tellers * service_rate

# empirical overload probability
p_exceed_emp = np.mean(arrivals > capacity)
```

Staffing decisions

- $np.random.poisson(lam, size = hours)$ simulates hourly counts from Poisson
- $sample\_mean$ and $sample\_var$ are the estimates of mean and variance.
- p_exceed_emp estimates the probability an hour's arrivals exceed the service capacity (= $tellers \times service\_rate$).
- $poisson.ppf()$ stands for the **Percent Point Function**
  - It is the inverse of the cumulative distribution function (CDF).
  - Find number of tellers k, such that $P(X \leq k) = 1 - \alpha = 0.95$
- The three plots show
  - (a) histogram of counts
  - (b) running mean (Law of Large Numbers)
  - (c) running variance (Poisson has Var ≈ Mean).
- $np.random.seed(1)$ makes all random operations reproducible.
  - $seed$ ensures you get the same random sequence every time you run the code

Bank arrivals

```
# theoretical capacity for a target risk alpha (e.g., alpha=0.05)
alpha = 0.05
k_needed = int(poisson.ppf(1-alpha, lam))  # smallest k s.t. P(N <= k) >= 1-alpha

# plots
plt.figure(); plt.hist(arrivals, bins=range(0, max(arrivals)+2)); plt.title("Hourly arrivals (hist)"); plt.xlabel("arrivals/hr")
plt.figure(); plt.plot(np.cumsum(arrivals)/np.arange(1, hours+1)); plt.title("Running mean")
plt.figure(); plt.plot([arrivals[:i].var(ddof=0) for i in range(1, hours+1)]); plt.title("Running variance")
plt.show()

print("sample_mean, sample_var, p_exceed_emp, k_needed:", sample_mean, sample_var, p_exceed_emp, k_needed)
```

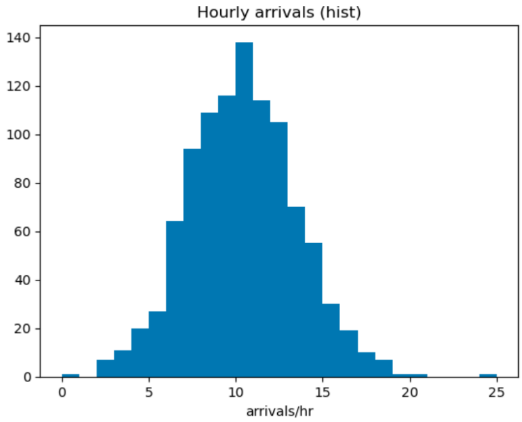
$sample\_mean, sample\_var, p\_exceed\_emp, k\_needed$: 9.907 9.688350999999999 0.667 15

Observations and Insights

- **Histogram**: shape is roughly Poisson/peaked near λ (here ≈ 10). The right tail shows risk of high arrival hours.
- Running mean: should converge toward λ as sample size increases — a visual demonstration of the Law of Large Numbers.
- Running variance: for Poisson, variance $\approx$ mean; the running variance should stabilize near the mean.
- **Exercise**:
  - Change *tellers* to 15 and observe *p_exceed_emp*
  - Change λ to 5, 15 and observe how *k_needed* changes.
  - Choose $\alpha = 0.01$ and compute the required capacity (*k_needed*)

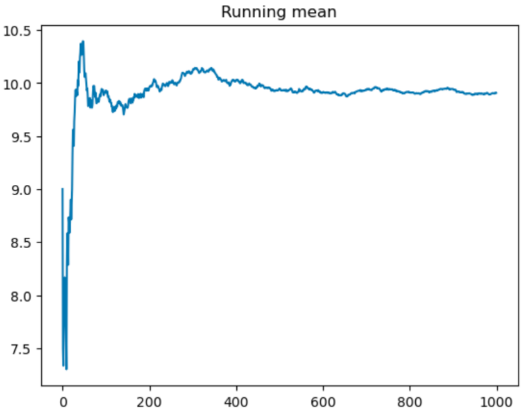
Observations

Histogram: shape is roughly Poisson/peaked near λ (here ≈ 10). The right tail shows risk of high arrival hours.



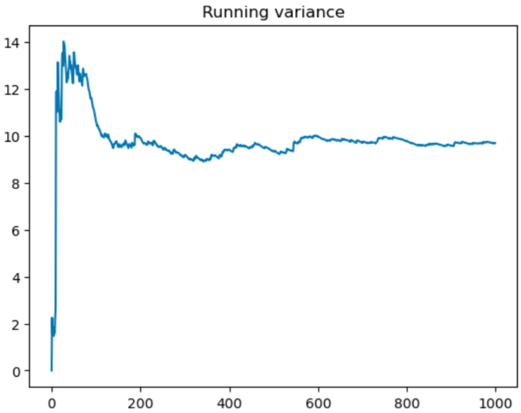
Observations

Running mean: should converge toward λ as sample size increases — a visual demonstration of the Law of Large Numbers.



Observations

Running variance: for Poisson, variance ≈ mean; the running variance should stabilize near the mean.



Betting Games

Betting Games

There are two games: A and B

- Game A: Outcomes: $\{1, -1\}$; Probability: $\{0.5, 0.5\}$
- Game B: Outcomes: $\{4, -1\}$; Probability: $\{0.3, 0.7\}$
- From the results:
  - Game A: mean $\approx$ 0, std $\approx$ 1, P(loss) $\approx$ 0.5.
  - Game B: mean $\approx$ 0.5, std $\approx$ 2.29, P(loss) $\approx$ 0.7.
- **Insights**
  - Game B has a higher long-run expected payoff but is riskier
  - If the player cares about long-run average (large number of plays) and can tolerate risk (variance), pick higher mean (Game B).
  - If the player is risk-averse or must avoid short-term losses, prefer lower variance even with lower EV (Game A).

Betting Games

```python
# Betting games - compare Expected Value and variance
import numpy as np
from math import sqrt
np.random.seed(1)
trials = 100000

# Game A: fair +/-1
gameA = np.random.choice([1, -1], size=trials)

# Game B: 30% win +4, 70% lose -1  => EV = 0.3*4 + 0.7*(-1) = 0.5
gameB = np.random.choice([4, -1], size=trials, p=[0.3, 0.7])

meanA, varA = gameA.mean(), gameA.var(ddof=0)
meanB, varB = gameB.mean(), gameB.var(ddof=0)
p_loss_A = np.mean(gameA < 0)
p_loss_B = np.mean(gameB < 0)

print("Game A: mean, std, P(loss) = ", meanA, sqrt(varA), p_loss_A)
print("Game B: mean, std, P(loss) = ", meanB, sqrt(varB), p_loss_B)
```

Reliability using conditional density

Reliability from Conditional Density

Reliability example

- Consider lifetimes of two machine components A and B together
- A and B are independent exponential lifetimes.
- But once we “condition” on A ?
- **Exercise**: Make B depend on A (B = 0.5*A + noise)
  - $B = 0.5 * A + np.random.normal(0, 1, N)$
- Observations:
  - When independent : conditional density ≈ marginal density (no change).
  - If they were dependent, the conditional PDF would shift or narrow.

Reliability


```python
from scipy.stats import expon

np.random.seed(2)
N = 10000
# lifetimes in hours ~ exponential
A = expon.rvs(scale=5, size=N)   # mean 5h
B = expon.rvs(scale=3, size=N)   # mean 3h
# system fails when both fail → joint lifetimes (for study)
plt.figure(figsize=(5,5))
plt.scatter(A, B, alpha=0.3)
plt.xlabel("Component A lifetime (h)")
plt.ylabel("Component B lifetime (h)")
plt.title("Joint sample from exponential lifetimes")
plt.show()

# conditional density estimate: P(B|A in [2,3])
mask = (A > 2) & (A < 3)
plt.figure()
plt.hist(B[mask], bins=50, density=True, alpha=0.7)
plt.title("Conditional density: lifetime of B | A∈[2,3]")
plt.xlabel("B lifetime (h)")
plt.ylabel("Density"); plt.show()
```

Reliability

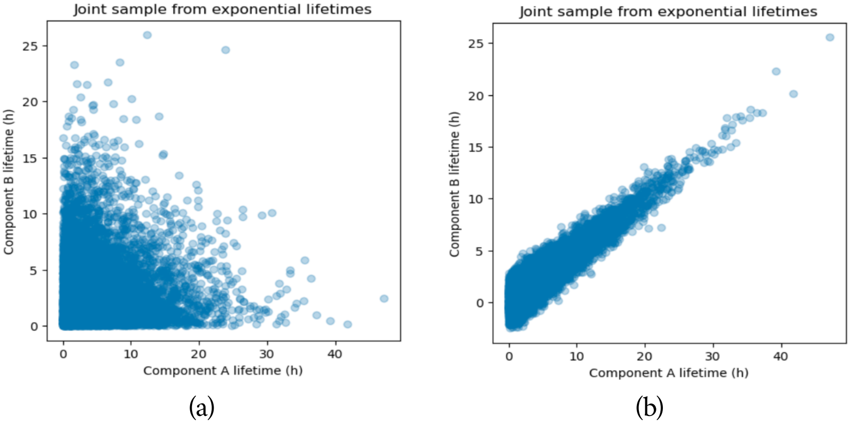
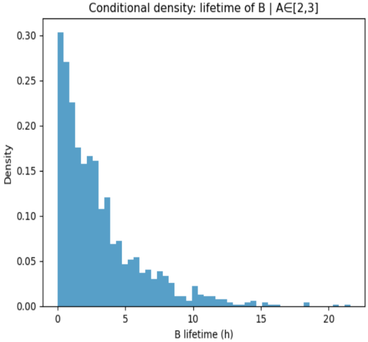
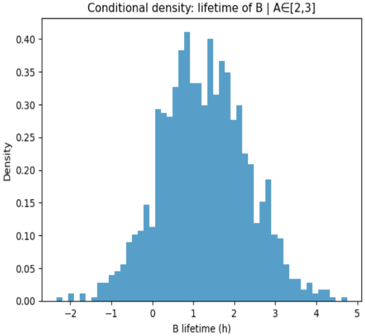


Figure: (a) Joint samples (A and B Independent) (b) Joint samples (B dependent on A)



(a)



(b)

Figure: (a) Conditional density B/A (A and B Independent) (b) Conditional density B/A (B dependent on A)

If they are dependent, the conditional PDF would shift or narrow.

Sum of Random variables

Sum of Random variables

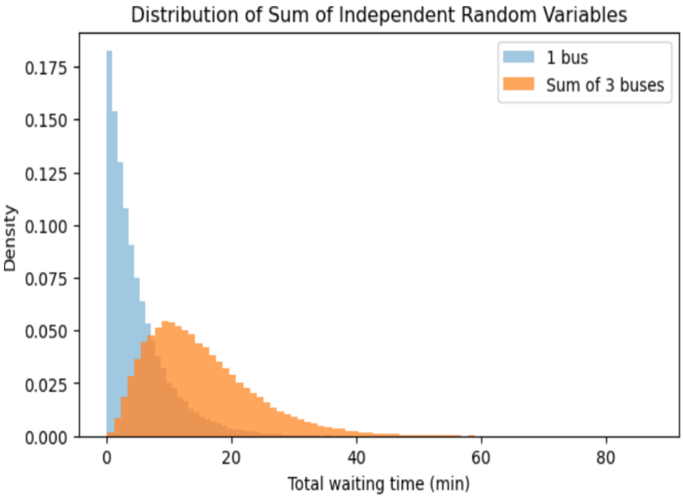
Consider waiting time for each bus as a random variable

- Each bus wait ≈ Exponential (λ=1/5).
- Sum of 3 such i.i.d. exponentials : Gamma(3, 5) distribution (smoother, more symmetric).
- Insights: Adding independent random variables **smooths** and **spreads** distributions.
- As we sum more terms $\longrightarrow$ shape tends toward normal (Central Limit Theorem)
- **Exercise**: Change number of buses to 10/20/30 and watch distribution become bell-shaped.

Sum of 3 Random variables

```
from scipy.stats import expon
np.random.seed(3)

N = 100000
bus1 = expon.rvs(scale=5, size=N)
bus2 = expon.rvs(scale=5, size=N)
bus3 = expon.rvs(scale=5, size=N)
total_time = bus1 + bus2 + bus3

plt.figure(figsize=(7,4))
plt.hist(bus1, bins=80, density=True, alpha=0.4, label='1 bus')
plt.hist(total_time, bins=80, density=True, alpha=0.7, label='Sum of 3 buses')
plt.xlabel("Total waiting time (min)"); plt.ylabel("Density")
plt.title("Distribution of Sum of Independent Random Variables")
plt.legend(); plt.show()

print("E[1 bus] =", round(bus1.mean(),2))
print("E[3 buses] =", round(total_time.mean(),2))
print("Var[1 bus] =", round(bus1.var(),2))
print("Var[3 buses] =", round(total_time.var(),2))
```

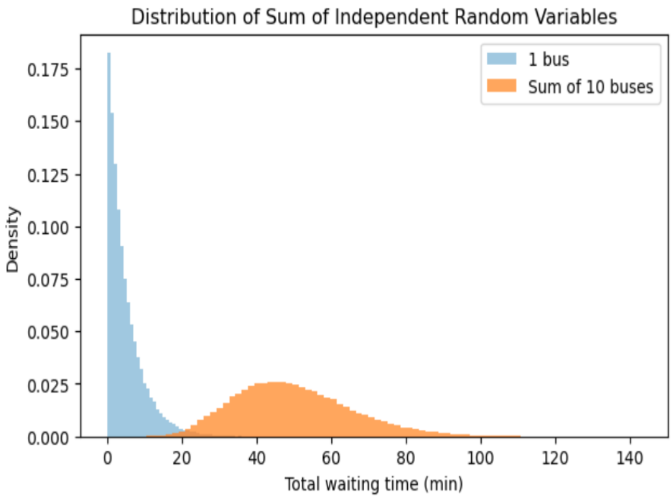
Sum of 10 Random variables



Sum of 10 Random variables


```python
from scipy.stats import expon
np.random.seed(3)

N = 100000
bus1 = expon.rvs(scale=5, size=N)
bus2 = expon.rvs(scale=5, size=N)
bus3 = expon.rvs(scale=5, size=N)
bus4 = expon.rvs(scale=5, size=N)
bus5 = expon.rvs(scale=5, size=N)
bus6 = expon.rvs(scale=5, size=N)
bus7 = expon.rvs(scale=5, size=N)
bus8 = expon.rvs(scale=5, size=N)
bus9 = expon.rvs(scale=5, size=N)
bus10 = expon.rvs(scale=5, size=N)

total_time = bus1 + bus2 + bus3+bus4+bus5+bus6+bus7+bus8+bus9+bus10

plt.figure(figsize=(7,4))
plt.hist(bus1, bins=80, density=True, alpha=0.4, label='1 bus')
plt.hist(total_time, bins=80, density=True, alpha=0.7, label='Sum of 10 buses')
plt.xlabel("Total waiting time (min)"); plt.ylabel("Density")
plt.title("Distribution of Sum of Independent Random Variables")
plt.legend(); plt.show()

print("E[1 bus] =", round(bus1.mean(),2))
print("E[3 buses] =", round(total_time.mean(),2))
print("Var[1 bus] =", round(bus1.var(),2))
print("Var[3 buses] =", round(total_time.var(),2))
```

Sum of 10 Random variables



Acknowledge various sources for the images.

Thankyou