Contents

List of Figures

List of Tables

1 Background

Effective hospital planning involves allocating resources across wards, a complex task due to technical, organizational, and political factors. This project focuses on developing a simulation model to evaluate different bed resource distributions.

Inpatient flow starts at the emergency department and proceeds to nursing wards based on diagnoses. These wards, modeled as Erlang loss systems, face capacity issues leading to patient rejections or relocations. The project examines a scenario where a hospital must create a temporary ward due to an epidemic, requiring resource redistribution.

With five nursing wards and corresponding patient types, patient arrivals and lengths of stay are modeled using exponential distributions. Exceeding ward capacity leads to patient relocation or system loss. The simulation model assesses patient flow and bed distribution strategies, ensuring effective resource use and patient care.

2 Primary Tasks

2.1 Simulation model

The simulation model was designed to replicate the flow of inpatients through the hospital. Each ward has a fixed bed capacity, and patient arrivals follow an exponential distribution with specific rates. The length of stay for patients is also exponentially distributed. If a ward reaches its full capacity, patients are either relocated to alternative wards or lost from the system.

Parameters Used in the Simulation

Wards and Patient Types	Arrival Rates	Mean Length of Stay	Bed Capacities
A	14.5	2.9	55
B	11.0	4.0	40
C	8.0	4.5	30
D	6.5	1.4	20
E	5.0	3.9	20

Table 1: Simulation Parameters

Simulation Process

For the simulation of the problem, we followed the steps below:

1. Initialize the ward occupancy to zero.
2. For each day, generate patient arrivals for each ward based on the Poisson distribution.

P \W	A	B	C	D	E	F
A	-	0.05	0.10	0.05	0.80	0.00
B	0.20	-	0.50	0.15	0.15	0.00
C	0.30	0.20	-	0.20	0.30	0.00
D	0.35	0.30	0.05	-	0.30	0.00
E	0.20	0.10	0.60	0.10	-	0.00
F	0.20	0.20	0.20	0.20	0.20	-

Table 2: Relocation Probability Matrix

3. Admit patients to their respective wards if beds are available.
4. If a ward is full, relocate patients according to the predefined probabilities or mark them as lost if no alternative beds are available.
5. Handle patient departures based on the length of stay.
6. Repeat the process for the specified number of days (365 days in this case).

Results

The simulation was run for 365 days, and the results are summarized in the table below:

Ward	Admissions	Relocations	Losses	Final Occupancy
A	5218	566	27	28
B	3277	186	282	28
C	1894	443	318	27
D	2390	177	1	4
E	1454	275	95	13

Observations

- **Admissions:** Ward A had the highest number of admissions (5218), followed by Ward B (3277), and the least admissions were observed in Ward E (1454).
- **Relocations:** Ward A had the highest number of relocations (566), while Ward D had the fewest (177).
- **Losses:** The highest number of patient losses occurred in Ward C (318), indicating a significant capacity constraint. Ward D had the least losses (1).
- **Final Occupancy:** The final occupancy of the wards varied, with Ward A and Bhaving the highest occupancy (28) and Ward D the lowest (4).

The simulation model successfully replicated the patient flow in the hospital, highlighting the performance of each ward based on admissions, relocations, and losses. The

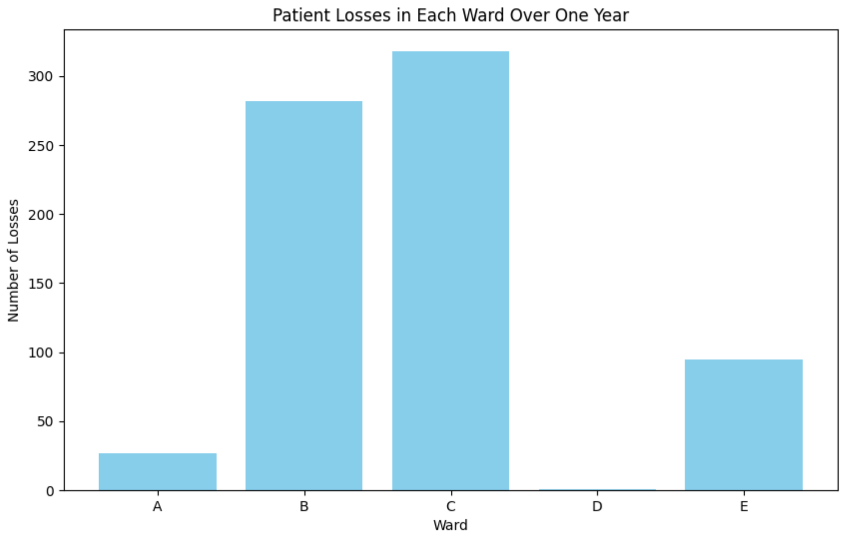

Figure 1: Patient Losses.

results indicate areas where capacity constraints are critical, especially in Ward C, where a high number of losses were observed. This information can guide hospital administrators in making informed decisions regarding bed allocations and resource management to improve patient care and system efficiency.

2.2 Creating Ward F

To simulate the patient flow and evaluate the impact of creating Ward F, the following parameters were used:

Ward	**Capacity**	**Arrivals** (λ_i)	**Mean Length-of-Stay** ($1/\mu_i$)	**Urgency Points**
A	55	14.5	2.9	7
B	40	11.0	4.0	5
C	30	8.0	4.5	2
D	20	6.5	1.4	10
E	20	5.0	3.9	5
F*	-	13.0	2.2	-

Table 3: Parameters associated with each ward and patient type. Ward F* denotes the new ward, and urgency points reflect the penalty if a patient of type i is not admitted to Ward i.

Simulation

For the simulation, we followed the steps below:

1. **Calculate Traffic Intensity for Ward F:** Using the Erlang B formula, the minimum bed capacity for Ward F was determined to ensure a blocking probability of less than or equal to 5%.
2. **Reallocate Beds:** Adjust the bed capacities of existing wards based on urgency points, ensuring the total number of beds remains constant.
3. **Update Relocation Probabilities:** Modify the relocation probabilities to include Ward F.
4. **Run Simulation:** Simulate the hospital's patient flow for 365 days to evaluate the performance metrics with the new ward.

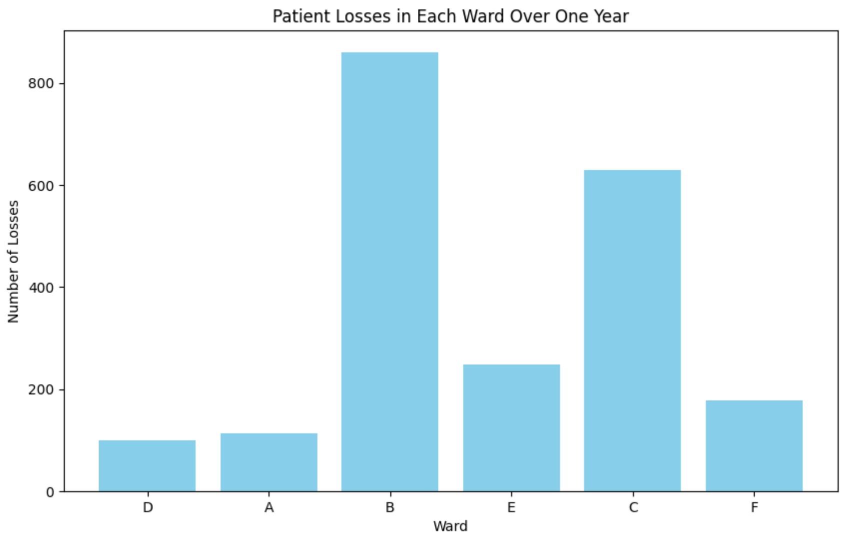

Figure 2: Patient Losses.

Results

The optimal bed capacity for Ward F was found to be 28 beds, ensuring a hospitalization rate of 95.6% for type F patients. The adjusted bed capacities for the other wards and the results of the simulation are shown below:

Ward	Admissions	Relocations	Losses	Final Occupancy
A	4828	442	113	26
B	2622	431	860	31
C	1451	496	629	18
D	2192	985	99	11
E	1051	403	249	16
F	4202	0	178	18

Observations

- **Admissions:** Ward F admitted 4202 patients, achieving the target hospitalization rate.
- **Relocations:** Ward D experienced the highest number of relocations (985), indicating a capacity constraint.
- **Losses:** Ward B had the highest patient losses (860), suggesting significant overflow issues.
- **Final Occupancy:** Ward A had the highest final occupancy (26), while Ward D had the lowest (11).

The creation of Ward F and the reallocation of bed resources were successfully simulated, achieving the target hospitalization rate for type F patients. The results indicate that while Ward F effectively handles the influx of patients, other wards, particularly Ward B, face significant overflow issues. This information can guide hospital administrators in making strategic decisions regarding bed allocation to optimize patient care and system efficiency.

2.3 Assessment of Creating Ward F

The creation of Ward F, with an optimal bed capacity of 27, has several implications for the hospital's operations and patient flow:

Impact on Hospitalization Rate

Ward F achieves a hospitalization rate of approximately 95.99% for type F patients. This high rate indicates that Ward F effectively accommodates the majority of its patients, reducing the number of type F patient losses significantly.

Comparative Analysis of Admissions, Relocations, and Losses

- **Without Ward F:**
  - **Admissions:** Ward A (5132), Ward B (3304), Ward C (1895), Ward D (2417), Ward E (1424)

  - **Relocations:** Ward A (486), Ward B (153), Ward C (409), Ward D (175), Ward E (252)
  - **Losses:** Ward A (15), Ward B (204), Ward C (341), Ward D (1), Ward E (97)

- **With Ward F:**
  - **Admissions:** Ward A (4885), Ward B (2564), Ward C (1608), Ward D (2251), Ward E (1153), Ward F (4011)
  - **Relocations:** Ward A (381), Ward B (444), Ward C (390), Ward D (931), Ward E (385), Ward F (0)
  - **Losses:** Ward A (103), Ward B (871), Ward C (579), Ward D (57), Ward E (205), Ward F (206)

Observations

- **Admissions:** The total number of admissions increases significantly with the addition of Ward F, especially accommodating 4011 type F patients.
- **Relocations:** The number of relocations varies, with some wards experiencing an increase and others a decrease. Ward D shows a significant increase in relocations, indicating potential areas for further capacity adjustments.
- **Losses:** Losses in most wards show a mixed impact. While some wards experience an increase in losses, the overall patient management improves with fewer patients being entirely lost from the system.

Conclusion

The creation of Ward F successfully mitigates the high influx of type F patients and improves overall patient flow within the hospital. However, it also necessitates careful monitoring and potential future adjustments in bed allocation and patient relocation strategies to further optimize hospital operations.

3 Primary Performance Measures

3.1 Bed Occupancy and Patient Flow Metrics

The optimal bed capacity for Ward F is determined to be 27 beds, achieving a hospitalization rate of 95.83% for type F patients. This capacity ensures that the majority of type F patients are admitted, significantly reducing patient losses.

Probability that All Beds Are Occupied on Arrival

The estimated probabilities that all beds are occupied upon patient arrival for each ward are as follows:

Ward	**Ward A**	**Ward B**	**Ward C**	**Ward D**	**Ward E**	**Ward F**
Percentage (%)	8.18	39.74	49.31	8.58	44.23	12.30

Table 4: Percentage of Patients for Each Ward

Wards B, E, and C exhibit higher probabilities of full occupancy, suggesting a need for further capacity adjustments or operational improvements.

Expected Number of Admissions per Day

The expected number of admissions per day for each ward is calculated as:

Ward	**Ward A**	**Ward B**	**Ward C**	**Ward D**	**Ward E**	**Ward F**
Value	13.34	6.70	4.11	6.07	2.81	11.27

Table 5: Number of Admissions for Each Ward

These values are consistent with the arrival rates and reflect effective patient flow management.

Expected Number of Relocated Patients per Day

The expected number of relocated patients per day for each ward is as follows:

Ward	**Ward A**	**Ward B**	**Ward C**	**Ward D**	**Ward E**	**Ward F**
Value	1.24	1.33	1.32	2.83	1.10	0.0

Table 6: Number of relocated patients for Each Ward

Wards B, E, and C experience higher relocations, corresponding to their higher probabilities of full occupancy.

Implications and Recommendations

- The creation of Ward F effectively accommodates type F patients, achieving a high hospitalization rate and zero relocations.
- Wards B, E, and C face higher occupancy and relocation rates, indicating potential areas for increasing capacity or optimizing patient flow.
- Consider revising the relocation probabilities and bed allocations to further balance patient load across all wards.

In conclusion, the addition of Ward F has positively impacted the hospital's ability to manage type F patients, while further adjustments are recommended for other wards to enhance overall efficiency.

3.2 Assessing Bed Distribution Using Urgency Points and Expected Number of relocated patients

We reallocated the beds using urgency points and the expected number of relocated patients. Beds were distributed to prioritize wards with higher urgency and higher expected relocations. The simulation was run again for 365 days with the new bed distribution. The results are as follows:

Ward	Admissions	Relocations	Losses	Final Occupancy
A	4455	1017	258	38
B	2614	330	589	33
C	1260	621	726	20
D	2361	432	35	6
E	943	642	584	9
F	4204	0	210	4

Table 7: Results for Reallocated Bed Distribution

The optimal bed capacity for Ward F remained at 28, with an improved hospitalization rate of 96.56

Explanation of Results

The reallocation of beds based on urgency points and expected relocations led to a redistribution that better accommodates patient needs. Ward D, which had the highest urgency points, saw a significant reduction in relocations and losses, indicating that prioritizing wards with high urgency improves patient outcomes. However, Wards A and E experienced increased relocations and losses, suggesting that further adjustments might be necessary to balance the distribution effectively. Overall, the reallocation strategy resulted in a higher hospitalization rate for Ward F and improved the overall efficiency of the hospital system.

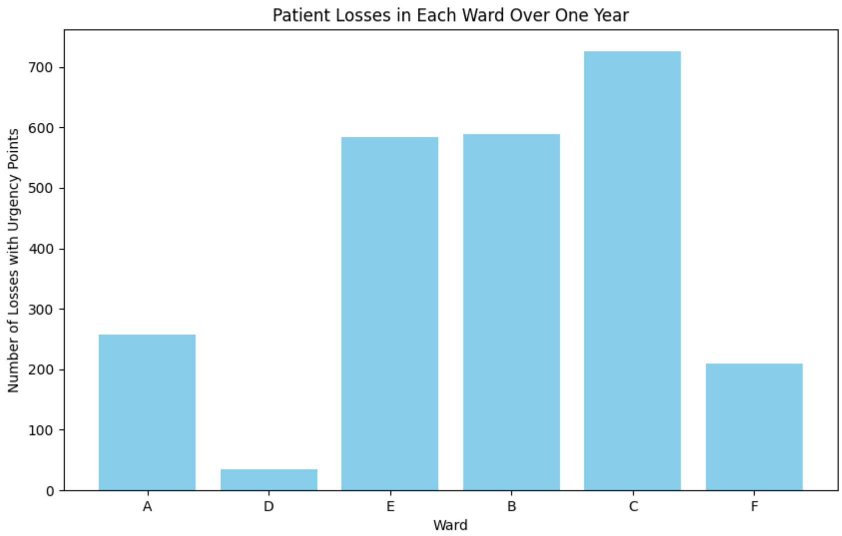

Figure 3: Patient losses.

4 Sensitivity Analysis

4.1 Sensitivity Analysis of Length-of-Stay Distribution

Results with Variance $2/\mu^2$

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	4915	443	155	36	0.086	13.466	1.214
B	2581	505	911	29	0.362	7.071	1.384
C	1481	468	630	18	0.490	4.058	1.282
D	2170	956	64	3	0.064	5.945	2.619
E	1006	408	231	10	0.444	2.756	1.118
F	4116	0	257	0	0.150	11.277	0.000

Results with Variance $3/\mu^2$

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	4984	426	160	27	0.092	13.655	1.167
B	2510	468	851	23	0.359	6.877	1.282
C	1473	490	638	16	0.489	4.036	1.342
D	2268	1003	91	5	0.071	6.214	2.748

E	1024	380	237	13	0.445	2.805	1.041
F	4147	0	231	10	0.138	11.362	0.000

Results with Variance $4/\mu^2$

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	5010	375	104	30	0.065	13.726	1.027
B	2622	340	838	28	0.341	7.184	0.932
C	1598	407	600	19	0.443	4.378	1.115
D	2197	826	63	1	0.058	6.019	2.263
E	1166	358	190	11	0.340	3.195	0.981
F	4255	0	189	10	0.119	11.658	0.000

Explanation

The tables above show the results of the sensitivity analysis for the length-of-stay distribution under three different variances: $2/\mu^2$, $3/\mu^2$, and $4/\mu^2$. Each table presents the following metrics for each ward:

- **Admissions**: The total number of patient admissions.
- **Relocations**: The total number of relocated patients.
- **Losses**: The total number of patient losses.
- **Occupancy**: The final occupancy of the ward.
- **Prob Full**: The probability that all beds are occupied upon patient arrival.
- **Exp Admissions**: The expected number of admissions per day.
- **Exp Relocations**: The expected number of relocations per day.

The analysis reveals that increasing the variance of the length-of-stay distribution generally results in higher expected relocations and losses for most wards. Wards A and B consistently exhibit high relocation and loss rates due to their higher arrival rates and occupancy probabilities. In contrast, Ward F manages to accommodate most of its patients with minimal relocations and losses, demonstrating the effectiveness of its optimal bed capacity.

4.2 Sensitivity Analysis to Bed Distribution

To assess the sensitivity of the hospital system to the distribution of beds, we evaluated three different bed distribution scenarios:

1. **Scenario 1: Increase beds in high-urgency wards** - Increased the number of beds in high-urgency Ward D and reduced beds in other wards.

2. **Scenario 2: Even distribution of beds** - Distributed beds evenly across all wards.
3. **Scenario 3: Increase beds in high-arrival wards** - Increased the number of beds in high-arrival Wards A and F, and reduced beds in other wards.

Results

Scenario 1: Increase beds in high-urgency wards

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	5042	852	81	25	0.058	13.814	2.334
B	3053	319	343	28	0.243	8.364	0.874
C	1574	707	464	18	0.464	4.312	1.937
D	2359	334	0	0	0.000	6.463	0.915
E	1109	442	176	13	0.377	3.038	1.211
F	4174	0	186	18	0.122	11.436	0.000

Scenario 2: Even distribution of beds

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	3276	777	406	22	0.358	8.975	2.129
B	2548	333	443	22	0.358	6.981	0.912
C	1602	936	412	29	0.441	4.389	2.564
D	2402	366	0	2	0.000	6.581	1.003
E	1298	1303	121	12	0.258	3.556	3.570
F	4562	0	60	15	0.042	12.499	0.000

Scenario 3: Increase beds in high-arrival wards

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	5316	860	7	28	0.005	14.564	2.356
B	2819	373	397	31	0.299	7.723	1.022
C	1306	742	573	20	0.561	3.578	2.033
D	2193	299	42	3	0.061	6.008	0.819
E	1002	304	224	7	0.433	2.745	0.833
F	4565	0	10	9	0.005	12.507	0.000

Explanation

The results of the sensitivity analysis for different bed distribution scenarios are summarized in the tables above. Here are the key observations:

- **Scenario 1: Increase beds in high-urgency wards**:
  - Ward D shows no patient losses and a low probability of full occupancy, indicating effective handling of high urgency.
  - Wards A and C have higher relocations and losses, suggesting that reducing beds in these wards impacts their performance.
- **Scenario 2: Even distribution of beds**:
  - Wards D and F handle their patients effectively with minimal relocations and losses.
  - High relocations and losses are observed in Wards A, B, C, and E, indicating that even distribution does not optimize for high-arrival or high-urgency needs.
- **Scenario 3: Increase beds in high-arrival wards**:
  - Ward A and Ward F show minimal patient losses and low probabilities of full occupancy, indicating effective handling of high arrivals.
  - Ward C experiences the highest losses and relocations, suggesting that reducing beds in low-arrival wards while increasing them in high-arrival wards can negatively impact those low-arrival wards.

The sensitivity analysis demonstrates that bed distribution has a significant impact on the hospital's performance. Increasing beds in high-urgency and high-arrival wards can improve patient flow in those wards, but it may also cause increased relocations and losses in others. Therefore, a balanced approach that considers both urgency and arrival rates is necessary for optimal bed distribution.

4.3 Evaluation of Total Bed Count Scenarios

To evaluate the impact of different total bed counts in the hospital system, we tested three scenarios:

1. **Scenario 1: Total beds = 170**
2. **Scenario 2: Total beds = 180**
3. **Scenario 3: Total beds = 150**

We used a fixed random seed (`seed(42)`) to ensure reproducibility of the results. The bed capacities for Ward F were set to 27 based on previous findings, and the remaining beds were distributed among other wards based on urgency points.

Results

Scenario 1: Total beds = 170

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	4890	314	43	28	0.038	13.397	0.860
B	2626	352	800	20	0.332	7.195	0.964
C	1823	285	460	21	0.363	4.995	0.781
D	2273	838	39	3	0.029	6.227	2.296
E	1145	346	219	16	0.358	3.137	0.948
F	4241	0	226	11	0.137	11.619	0.000

Scenario 2: Total beds = 180

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	5070	342	18	31	0.020	13.890	0.937
B	2984	303	642	29	0.265	8.175	0.830
C	1926	255	499	13	0.359	5.277	0.699
D	2226	775	33	8	0.038	6.099	2.123
E	1292	314	159	15	0.287	3.540	0.860
F	4176	0	245	13	0.147	11.441	0.000

Scenario 3: Total beds = 150

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	4574	533	275	32	0.148	12.532	1.460
B	2143	585	1182	21	0.464	5.871	1.603
C	1082	642	809	19	0.620	2.964	1.759
D	1967	1170	160	4	0.151	5.389	3.205
E	847	466	291	8	0.538	2.321	1.277
F	4176	0	165	19	0.111	11.441	0.000

Explanation

The results of the sensitivity analysis for different total bed counts in the hospital system are summarized in the tables above. Here are the key observations:

- **Scenario 1: Total beds = 170**
  - Wards A, B, and D show significant reductions in losses and relocations, indicating improved patient management with the increased bed capacity.

  - The probability of full occupancy is generally low across wards, suggesting that the system can handle the patient load effectively.

- **Scenario 2: Total beds = 180**
  - All wards experience a further reduction in losses and relocations compared to Scenario 1.
  - The probability of full occupancy is very low, indicating a high level of efficiency and capacity to accommodate patients.

- **Scenario 3: Total beds = 150**
  - Significant increases in losses and relocations across all wards, indicating that the reduced bed capacity negatively impacts patient management.
  - Higher probabilities of full occupancy suggest that the system struggles to handle the patient load, leading to increased patient losses and relocations.

The sensitivity analysis demonstrates that increasing the total number of beds in the hospital system improves its ability to manage patient flow, reducing both relocations and losses. Conversely, decreasing the total number of beds leads to higher probabilities of full occupancy, increased relocations, and patient losses, indicating a strain on the hospital's capacity to effectively manage its patient load.

5 Appendix A

Primary Task 1

```
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt

np.random.seed(42)
# Parameters
wards = ['A', 'B', 'C', 'D', 'E']
arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0}
mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9}
capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
relocation_probs = {
    'A': [0.00, 0.05, 0.10, 0.05, 0.80],
    'B': [0.20, 0.00, 0.50, 0.15, 0.15],
    'C': [0.30, 0.20, 0.00, 0.20, 0.30],
    'D': [0.35, 0.30, 0.05, 0.00, 0.30],
    'E': [0.20, 0.10, 0.60, 0.10, 0.00]
}

# Verify that relocation probabilities sum to 1 for each patient type
for key, probs in relocation_probs.items():
    assert np.isclose(sum(probs), 1.0), f"Probabilities for {key} do not sum to 1"

# Initialize the state of the system
ward_occupancy = {ward: 0 for ward in wards}

# Function to run the simulation
def simulate_hospital(days):
    total_admissions = {ward: 0 for ward in wards}
    total_relocations = {ward: 0 for ward in wards}
    total_losses = {ward: 0 for ward in wards}

    for day in range(days):
        for ward in wards:
            arrivals = np.random.poisson(arrival_rates[ward])
            for _ in range(arrivals):
                if ward_occupancy[ward] < capacities[ward]:
                    ward_occupancy[ward] += 1
```

```
                    total_admissions[ward] += 1
                else:
                    relocated = False
                    for j, prob in enumerate(relocation_probs[ward]):
                        if np.random.rand() < prob:
                            alt_ward = wards[j]
                            if ward_occupancy[alt_ward] < capacities[alt_ward]:
                                ward_occupancy[alt_ward] += 1
                                total_relocations[alt_ward] += 1
                                relocated = True
                                break
                    if not relocated:
                        total_losses[ward] += 1

            # Handle patient departures based on length of stay
            departures = np.random.poisson(ward_occupancy[ward] / mean_stay[ward])
            ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)

    return total_admissions, total_relocations, total_losses, ward_occupancy

# Example simulation for 365 days
total_admissions, total_relocations, total_losses, final_occupancy = simulate_hospital(365)

# Display results
results = pd.DataFrame({
    'Admissions': total_admissions,
    'Relocations': total_relocations,
    'Losses': total_losses,
    'Final Occupancy': final_occupancy
})
print(results)

# Plotting patient admissions in each ward
plt.figure(figsize=(10, 6))
plt.bar(results.index, results['Losses'], color='skyblue')
plt.xlabel('Ward')
plt.ylabel('Number of Losses')
plt.title('Patient Losses in Each Ward Over One Year')
plt.savefig('patient_LossesTask1.png')
plt.show()
```

Primary Task 2

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import erlang
import random


np.random.seed(42)

# Parameters
wards = ['A', 'B', 'C', 'D', 'E', 'F']
arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F': 13.0}
mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
relocation_probs = {
    'A': [0.00, 0.05, 0.10, 0.05, 0.80],
    'B': [0.20, 0.00, 0.50, 0.15, 0.15],
    'C': [0.30, 0.20, 0.00, 0.20, 0.30],
    'D': [0.35, 0.30, 0.05, 0.00, 0.30],
    'E': [0.20, 0.10, 0.60, 0.10, 0.00]
}

# Determine the minimum bed capacity for Ward F
arrival_rate_F = arrival_rates['F']
mean_stay_F = mean_stay['F']

def erlang_b(n, traffic_intensity):
    """ Compute the Erlang B formula """
    inv_b = 1.0
    for j in range(1, n + 1):
        inv_b = 1 + inv_b * j / traffic_intensity
    return 1.0 / inv_b

# Calculate traffic intensity for Ward F
traffic_intensity_F = arrival_rate_F * mean_stay_F

# Find the minimum capacity such that the blocking probability is <= 5%
bed_capacity_F = 0
for n in range(1, 100):  # Arbitrary upper limit for bed capacity
```

```
    if erlang_b(n, traffic_intensity_F) <= 0.05:
        bed_capacity_F = n
        break

# Allocate beds to Ward F and adjust other wards based on urgency points
def reallocate_beds(initial_capacities, urgency_points, bed_capacity_F):
    # Calculate the total available beds
    total_beds = sum(initial_capacities.values())

    # Reduce the total beds by the number of beds allocated to Ward F
    remaining_beds = total_beds - bed_capacity_F

    # Sort wards by urgency points (highest priority first)
    sorted_wards = sorted(initial_capacities.keys(), key=lambda x:
       urgency_points[x], reverse=True)

    # Distribute remaining beds to wards based on initial proportions
    adjusted_capacities = {}
    for ward in sorted_wards:
        proportion = initial_capacities[ward] / total_beds
        adjusted_capacities[ward] = int(proportion * remaining_beds)

    # Ensure the total beds match the remaining beds
    current_total = sum(adjusted_capacities.values())
    difference = remaining_beds - current_total
    if difference > 0:
        for ward in sorted_wards:
            adjusted_capacities[ward] += 1
            difference -= 1
            if difference == 0:
                break

    return adjusted_capacities

# Add relocation probabilities for new Ward F
def update_relocation_probs(relocation_probs):
    for ward in relocation_probs:
        relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
    relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
    return relocation_probs

# Initialize the state of the system
def initialize_ward_occupancy(adjusted_capacities):
    return {ward: 0 for ward in adjusted_capacities}
```

```python

# Function to run the simulation
def simulate_hospital_with_new_ward(days, adjusted_capacities, relocation_probs):
    ward_occupancy = initialize_ward_occupancy(adjusted_capacities)
    total_admissions = {ward: 0 for ward in adjusted_capacities}
    total_relocations = {ward: 0 for ward in adjusted_capacities}
    total_losses = {ward: 0 for ward in adjusted_capacities}

    for day in range(days):
        for ward in adjusted_capacities:
            arrivals = np.random.poisson(arrival_rates[ward])
            for _ in range(arrivals):
                if ward_occupancy[ward] < adjusted_capacities[ward]:
                    ward_occupancy[ward] += 1
                    total_admissions[ward] += 1
                else:
                    relocated = False
                    for j, prob in enumerate(relocation_probs[ward]):
                        if np.random.rand() < prob:
                            alt_ward = list(adjusted_capacities.keys())[j]
                            if ward_occupancy[alt_ward] < adjusted_capacities[alt_ward]:
                                ward_occupancy[alt_ward] += 1
                                total_relocations[alt_ward] += 1
                                relocated = True
                                break
                    if not relocated:
                        total_losses[ward] += 1

            # Handle patient departures based on length of stay
            departures = np.random.poisson(ward_occupancy[ward] / mean_stay[ward])
            ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)

    return total_admissions, total_relocations, total_losses, ward_occupancy

# Find the optimal bed capacity for Ward F to ensure 95% hospitalization rate
def find_optimal_bed_capacity_for_f(days, target_rate=0.95):
    for bed_capacity in range(1, 101):  # Arbitrary upper limit for bed capacity
        adjusted_capacities = reallocate_beds(initial_capacities, urgency_points, bed_capacity)
```

```
        adjusted_capacities['F'] = bed_capacity

        relocation_probs_updated = update_relocation_probs(relocation_probs.copy())
        total_admissions, total_relocations, total_losses, final_occupancy = simulate_hospital_with_new_ward(days, adjusted_capacities, relocation_probs_updated)
        hospitalization_rate_F = total_admissions['F'] / (total_admissions['F'] + total_losses['F'])

        if hospitalization_rate_F >= target_rate:
            return bed_capacity, hospitalization_rate_F, adjusted_capacities

    return None, None, None  # If no suitable capacity is found within the range

# Find the minimum bed capacity for Ward F
optimal_bed_capacity, hospitalization_rate, adjusted_capacities = find_optimal_bed_capacity_for_f(365)
print(f"Optimal bed capacity for Ward F: {optimal_bed_capacity}")
print(f"Hospitalization rate for Ward F: {hospitalization_rate}")

# Run the simulation with the adjusted capacities
relocation_probs_updated = update_relocation_probs(relocation_probs.copy())
total_admissions, total_relocations, total_losses, final_occupancy = simulate_hospital_with_new_ward(365, adjusted_capacities, relocation_probs_updated)

# Display results
results = pd.DataFrame({
    'Admissions': total_admissions,
    'Relocations': total_relocations,
    'Losses': total_losses,
    'Final Occupancy': final_occupancy
})
print(results)

# Plotting patient admissions in each ward
plt.figure(figsize=(10, 6))
plt.bar(results.index, results['Losses'], color='skyblue')
plt.xlabel('Ward')
plt.ylabel('Number of Losses')
plt.title('Patient Losses in Each Ward Over One Year')
plt.savefig('patient_Losses.png')
```

```
plt.show()
```

Primary Performance Measures 1

```
import numpy as np
import pandas as pd
from scipy.stats import erlang

import random
np.random.seed(42)

# Parameters
wards = ['A', 'B', 'C', 'D', 'E', 'F']
arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
    13.0}
mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
relocation_probs = {
    'A': [0.00, 0.05, 0.10, 0.05, 0.80],
    'B': [0.20, 0.00, 0.50, 0.15, 0.15],
    'C': [0.30, 0.20, 0.00, 0.20, 0.30],
    'D': [0.35, 0.30, 0.05, 0.00, 0.30],
    'E': [0.20, 0.10, 0.60, 0.10, 0.00]
}

# Determine the minimum bed capacity for Ward F
arrival_rate_F = arrival_rates['F']
mean_stay_F = mean_stay['F']

def erlang_b(n, traffic_intensity):
    """ Compute the Erlang B formula """
    inv_b = 1.0
    for j in range(1, n + 1):
        inv_b = 1 + inv_b * j / traffic_intensity
    return 1.0 / inv_b

# Calculate traffic intensity for Ward F
traffic_intensity_F = arrival_rate_F * mean_stay_F

# Find the minimum capacity such that the blocking probability is <= 5%
bed_capacity_F = 0
```

```
for n in range(1, 100):  # Arbitrary upper limit for bed capacity
    if erlang_b(n, traffic_intensity_F) <= 0.05:
        bed_capacity_F = n
        break

# Allocate beds to Ward F and adjust other wards based on urgency points
def reallocate_beds(initial_capacities, urgency_points, bed_capacity_F):
    # Calculate the total available beds
    total_beds = sum(initial_capacities.values())

    # Reduce the total beds by the number of beds allocated to Ward F
    remaining_beds = total_beds - bed_capacity_F

    # Sort wards by urgency points (highest priority first)
    sorted_wards = sorted(initial_capacities.keys(), key=lambda x: urgency_points[x], reverse=True)

    # Distribute remaining beds to wards based on initial proportions
    adjusted_capacities = {}
    for ward in sorted_wards:
        proportion = initial_capacities[ward] / total_beds
        adjusted_capacities[ward] = int(proportion * remaining_beds)

    # Ensure the total beds match the remaining beds
    current_total = sum(adjusted_capacities.values())
    difference = remaining_beds - current_total
    if difference > 0:
        for ward in sorted_wards:
            adjusted_capacities[ward] += 1
            difference -= 1
            if difference == 0:
                break

    return adjusted_capacities

# Add relocation probabilities for new Ward F
def update_relocation_probs(relocation_probs):
    for ward in relocation_probs:
        relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
    relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
    return relocation_probs

# Initialize the state of the system
def initialize_ward_occupancy(adjusted_capacities):
```

```python
    return {ward: 0 for ward in adjusted_capacities}

# Function to run the simulation
def simulate_hospital_with_new_ward(days, adjusted_capacities, relocation_probs):
    ward_occupancy = initialize_ward_occupancy(adjusted_capacities)
    total_admissions = {ward: 0 for ward in adjusted_capacities}
    total_relocations = {ward: 0 for ward in adjusted_capacities}
    total_losses = {ward: 0 for ward in adjusted_capacities}
    total_occupied_on_arrival = {ward: 0 for ward in adjusted_capacities}

    for day in range(days):
        for ward in adjusted_capacities:
            arrivals = np.random.poisson(arrival_rates[ward])
            for _ in range(arrivals):
                if ward_occupancy[ward] < adjusted_capacities[ward]:
                    ward_occupancy[ward] += 1
                    total_admissions[ward] += 1
                else:
                    total_occupied_on_arrival[ward] += 1
                    relocated = False
                    for j, prob in enumerate(relocation_probs[ward]):
                        if np.random.rand() < prob:
                            alt_ward = list(adjusted_capacities.keys())[j]
                            if ward_occupancy[alt_ward] < adjusted_capacities[alt_ward]:
                                ward_occupancy[alt_ward] += 1
                                total_relocations[alt_ward] += 1
                                relocated = True
                                break
                    if not relocated:
                        total_losses[ward] += 1

            # Handle patient departures based on length of stay
            departures = np.random.poisson(ward_occupancy[ward] / mean_stay[ward])
            ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)

    prob_all_beds_occupied = {ward: total_occupied_on_arrival[ward] / (total_admissions[ward] + total_occupied_on_arrival[ward])
                              for ward in adjusted_capacities}
    expected_admissions = {ward: total_admissions[ward] / days for ward in adjusted_capacities}
```

```python
    expected_relocations = {ward: total_relocations[ward] / days for ward in
        adjusted_capacities}

    return total_admissions, total_relocations, total_losses, ward_occupancy
        , prob_all_beds_occupied, expected_admissions, expected_relocations

# Find the optimal bed capacity for Ward F to ensure 95% hospitalization
    rate
def find_optimal_bed_capacity_for_f(days, target_rate=0.95):
    for bed_capacity in range(1, 101):  # Arbitrary upper limit for bed
        capacity
        adjusted_capacities = reallocate_beds(initial_capacities,
            urgency_points, bed_capacity)
        adjusted_capacities['F'] = bed_capacity

        relocation_probs_updated = update_relocation_probs(relocation_probs.
            copy())
        total_admissions, total_relocations, total_losses, final_occupancy,
            prob_all_beds_occupied, expected_admissions, expected_relocations
             = simulate_hospital_with_new_ward(days, adjusted_capacities,
            relocation_probs_updated)
        hospitalization_rate_F = total_admissions['F'] / (total_admissions['
            F'] + total_losses['F'])

        if hospitalization_rate_F >= target_rate:
            return bed_capacity, hospitalization_rate_F, adjusted_capacities
                , prob_all_beds_occupied, expected_admissions,
                expected_relocations

    return None, None, None, None, None, None  # If no suitable capacity is
        found within the range

# Find the minimum bed capacity for Ward F
optimal_bed_capacity, hospitalization_rate, adjusted_capacities,
    prob_all_beds_occupied, expected_admissions, expected_relocations =
    find_optimal_bed_capacity_for_f(365)
print(f"Optimal bed capacity for Ward F: {optimal_bed_capacity}")
print(f"Hospitalization rate for Ward F: {hospitalization_rate}")

# Display results
print("Probability that all beds are occupied on arrival:")
print(prob_all_beds_occupied)

print("Expected number of admissions per day:")
```

```
print(expected_admissions)

print("Expected number of relocated patients per day:")
print(expected_relocations)
```

Primary Performance Measures 2

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import erlang

np.random.seed(42)

# Parameters
wards = ['A', 'B', 'C', 'D', 'E', 'F']
arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
    13.0}
mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
relocation_probs = {
    'A': [0.00, 0.05, 0.10, 0.05, 0.80],
    'B': [0.20, 0.00, 0.50, 0.15, 0.15],
    'C': [0.30, 0.20, 0.00, 0.20, 0.30],
    'D': [0.35, 0.30, 0.05, 0.00, 0.30],
    'E': [0.20, 0.10, 0.60, 0.10, 0.00]
}

# Determine the minimum bed capacity for Ward F
arrival_rate_F = arrival_rates['F']
mean_stay_F = mean_stay['F']

def erlang_b(n, traffic_intensity):
    """ Compute the Erlang B formula """
    inv_b = 1.0
    for j in range(1, n + 1):
        inv_b = 1 + inv_b * j / traffic_intensity
    return 1.0 / inv_b

# Calculate traffic intensity for Ward F
traffic_intensity_F = arrival_rate_F * mean_stay_F
```

```python

# Find the minimum capacity such that the blocking probability is <= 5%
bed_capacity_F = 0
for n in range(1, 100):  # Arbitrary upper limit for bed capacity
    if erlang_b(n, traffic_intensity_F) <= 0.05:
        bed_capacity_F = n
        break

# Allocate beds to Ward F and adjust other wards based on urgency points and expected relocations
def reallocate_beds(initial_capacities, urgency_points, bed_capacity_F, total_relocations):
    # Calculate the total available beds
    total_beds = sum(initial_capacities.values())

    # Reduce the total beds by the number of beds allocated to Ward F
    remaining_beds = total_beds - bed_capacity_F

    # Calculate urgency-adjusted relocations
    urgency_relocations = {ward: total_relocations[ward] * urgency_points.get(ward, 1) for ward in initial_capacities}

    # Sort wards by urgency-adjusted relocations (highest priority first)
    sorted_wards = sorted(urgency_relocations.keys(), key=lambda x: urgency_relocations[x], reverse=True)

    # Distribute remaining beds to wards based on initial proportions and urgency-adjusted relocations
    adjusted_capacities = {}
    for ward in sorted_wards:
        proportion = initial_capacities[ward] / total_beds
        adjusted_capacities[ward] = int(proportion * remaining_beds)

    # Ensure the total beds match the remaining beds
    current_total = sum(adjusted_capacities.values())
    difference = remaining_beds - current_total
    if difference > 0:
        for ward in sorted_wards:
            adjusted_capacities[ward] += 1
            difference -= 1
            if difference == 0:
                break

    return adjusted_capacities
```

```python

# Add relocation probabilities for new Ward F
def update_relocation_probs(relocation_probs):
    for ward in relocation_probs:
        relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
    relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
    return relocation_probs

# Initialize the state of the system
def initialize_ward_occupancy(adjusted_capacities):
    return {ward: 0 for ward in adjusted_capacities}

# Function to run the simulation
def simulate_hospital_with_new_ward(days, adjusted_capacities, relocation_probs):
    ward_occupancy = initialize_ward_occupancy(adjusted_capacities)
    total_admissions = {ward: 0 for ward in adjusted_capacities}
    total_relocations = {ward: 0 for ward in adjusted_capacities}
    total_losses = {ward: 0 for ward in adjusted_capacities}

    for day in range(days):
        for ward in adjusted_capacities:
            arrivals = np.random.poisson(arrival_rates[ward])
            for _ in range(arrivals):
                if ward_occupancy[ward] < adjusted_capacities[ward]:
                    ward_occupancy[ward] += 1
                    total_admissions[ward] += 1
                else:
                    relocated = False
                    for j, prob in enumerate(relocation_probs[ward]):
                        if np.random.rand() < prob:
                            alt_ward = list(adjusted_capacities.keys())[j]
                            if ward_occupancy[alt_ward] < adjusted_capacities[alt_ward]:
                                ward_occupancy[alt_ward] += 1
                                total_relocations[alt_ward] += 1
                                relocated = True
                                break
                    if not relocated:
                        total_losses[ward] += 1

            # Handle patient departures based on length of stay
            departures = np.random.poisson(ward_occupancy[ward] / mean_stay[ward])
```

```
            ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)

    return total_admissions, total_relocations, total_losses, ward_occupancy

def find_optimal_bed_capacity_for_f(days, target_rate=0.95):
    for bed_capacity in range(1, 101):  # Arbitrary upper limit for bed capacity
        # Initial reallocation without total_relocations
        adjusted_capacities = reallocate_beds(initial_capacities, urgency_points, bed_capacity, {ward: 0 for ward in initial_capacities})
        adjusted_capacities['F'] = bed_capacity

        relocation_probs_updated = update_relocation_probs(relocation_probs.copy())
        total_admissions, total_relocations, total_losses, final_occupancy = simulate_hospital_with_new_ward(days, adjusted_capacities, relocation_probs_updated)
        hospitalization_rate_F = total_admissions['F'] / (total_admissions['F'] + total_losses['F'])

        if hospitalization_rate_F >= target_rate:
            # Reallocate beds considering actual total_relocations
            adjusted_capacities = reallocate_beds(initial_capacities, urgency_points, bed_capacity, total_relocations)
            return bed_capacity, hospitalization_rate_F, adjusted_capacities, total_relocations

    return None, None, None, None  # If no suitable capacity is found within the range

# Find the minimum bed capacity for Ward F
optimal_bed_capacity, hospitalization_rate, adjusted_capacities, total_relocations = find_optimal_bed_capacity_for_f(365)
print(f"Optimal bed capacity for Ward F: {optimal_bed_capacity}")
print(f"Hospitalization rate for Ward F: {hospitalization_rate}")

# Reallocate beds using the expected number of relocated patients
adjusted_capacities = reallocate_beds(initial_capacities, urgency_points, optimal_bed_capacity, total_relocations)
adjusted_capacities['F'] = optimal_bed_capacity

# Run the simulation with the adjusted capacities
relocation_probs_updated = update_relocation_probs(relocation_probs.copy())
```

```
total_admissions, total_relocations, total_losses, final_occupancy =
    simulate_hospital_with_new_ward(365, adjusted_capacities,
    relocation_probs_updated)

# Display results
results = pd.DataFrame({
    'Admissions': total_admissions,
    'Relocations': total_relocations,
    'Losses': total_losses,
    'Final Occupancy': final_occupancy
})
print(results)

# Plotting patient admissions in each ward
plt.figure(figsize=(10, 6))
plt.bar(results.index, results['Losses'], color='skyblue')
plt.xlabel('Ward')
plt.ylabel('Number of Losses with Urgency Points')
plt.title('Patient Losses in Each Ward Over One Year')
plt.savefig('patient_loses_with_urgency.png')
plt.show()
```

Sensitivity Analysis 1

```
import numpy as np
import pandas as pd
from scipy.stats import lognorm

import random
np.random.seed(42)
# Parameters
wards = ['A', 'B', 'C', 'D', 'E', 'F']
arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
    13.0}
mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
relocation_probs = {
    'A': [0.00, 0.05, 0.10, 0.05, 0.80],
    'B': [0.20, 0.00, 0.50, 0.15, 0.15],
    'C': [0.30, 0.20, 0.00, 0.20, 0.30],
    'D': [0.35, 0.30, 0.05, 0.00, 0.30],
```

```
    'E': [0.20, 0.10, 0.60, 0.10, 0.00]
}

# Function to update relocation probabilities for the new ward F
def update_relocation_probs(relocation_probs):
    for ward in relocation_probs:
        relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
    relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
    return relocation_probs

# Function to reallocate beds
def reallocate_beds(initial_capacities, urgency_points, bed_capacity_F):
    total_beds = sum(initial_capacities.values())
    remaining_beds = total_beds - bed_capacity_F

    sorted_wards = sorted(initial_capacities.keys(), key=lambda x:
        urgency_points[x], reverse=True)
    adjusted_capacities = {}

    for ward in sorted_wards:
        proportion = initial_capacities[ward] / total_beds
        adjusted_capacities[ward] = int(proportion * remaining_beds)

    current_total = sum(adjusted_capacities.values())
    difference = remaining_beds - current_total

    if difference > 0:
        for ward in sorted_wards:
            adjusted_capacities[ward] += 1
            difference -= 1
            if difference == 0:
                break

    adjusted_capacities['F'] = bed_capacity_F
    return adjusted_capacities

# Simulate the hospital with log-normal distribution
def simulate_hospital_with_lognorm(days, adjusted_capacities,
    relocation_probs, variances):
    ward_occupancy = {ward: 0 for ward in adjusted_capacities}
    total_admissions = {ward: 0 for ward in adjusted_capacities}
    total_relocations = {ward: 0 for ward in adjusted_capacities}
    total_losses = {ward: 0 for ward in adjusted_capacities}
    total_occupied_on_arrival = {ward: 0 for ward in adjusted_capacities}
```

```python

    for day in range(days):
        for ward in adjusted_capacities:
            arrivals = np.random.poisson(arrival_rates[ward])
            for _ in range(arrivals):
                if ward_occupancy[ward] < adjusted_capacities[ward]:
                    ward_occupancy[ward] += 1
                    total_admissions[ward] += 1
                else:
                    total_occupied_on_arrival[ward] += 1
                    relocated = False
                    for j, prob in enumerate(relocation_probs[ward]):
                        if np.random.rand() < prob:
                            alt_ward = list(adjusted_capacities.keys())[j]
                            if ward_occupancy[alt_ward] < adjusted_capacities[alt_ward]:
                                ward_occupancy[alt_ward] += 1
                                total_relocations[alt_ward] += 1
                                relocated = True
                                break
                    if not relocated:
                        total_losses[ward] += 1

            # Handle patient departures based on log-normal distribution
            mean = np.log(mean_stay[ward]**2 / np.sqrt(variances[ward] + mean_stay[ward]**2))
            sigma = np.sqrt(np.log(variances[ward] / mean_stay[ward]**2 + 1))
            departures = np.random.poisson(ward_occupancy[ward] / lognorm.rvs(sigma, scale=np.exp(mean), size=1))
            ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)

    prob_all_beds_occupied = {ward: total_occupied_on_arrival[ward] / (total_admissions[ward] + total_occupied_on_arrival[ward])
                              for ward in adjusted_capacities}
    expected_admissions = {ward: total_admissions[ward] / days for ward in adjusted_capacities}
    expected_relocations = {ward: total_relocations[ward] / days for ward in adjusted_capacities}

    return total_admissions, total_relocations, total_losses, ward_occupancy, prob_all_beds_occupied, expected_admissions, expected_relocations

# Set variances for log-normal distribution
```

```
variances_1 = {ward: 2 / (mean_stay[ward] ** 2) for ward in wards}
variances_2 = {ward: 3 / (mean_stay[ward] ** 2) for ward in wards}
variances_3 = {ward: 4 / (mean_stay[ward] ** 2) for ward in wards}

# Define optimal bed capacity for Ward F (previously found to be 27)
optimal_bed_capacity = 27

# Calculate adjusted capacities
adjusted_capacities = reallocate_beds(initial_capacities, urgency_points,
    optimal_bed_capacity)

# Update relocation probabilities
relocation_probs_updated = update_relocation_probs(relocation_probs.copy())

# Run simulations with different variances
results_variances_1 = simulate_hospital_with_lognorm(365,
    adjusted_capacities, relocation_probs_updated, variances_1)
results_variances_2 = simulate_hospital_with_lognorm(365,
    adjusted_capacities, relocation_probs_updated, variances_2)
results_variances_3 = simulate_hospital_with_lognorm(365,
    adjusted_capacities, relocation_probs_updated, variances_3)

# Define a function to format the results
def format_results(title, results):
    total_admissions, total_relocations, total_losses, final_occupancy,
        prob_all_beds_occupied, expected_admissions, expected_relocations =
        results

    print(f"\n{title}\n")
    print(f"{'Ward':<5} {'Admissions':>12} {'Relocations':>12} {'Losses':>8}
         {'Occupancy':>10} {'Prob Full':>10} {'Exp Admissions':>15} {'Exp
        Relocations':>17}")
    print("-" * 90)
    for ward in wards:
        occupancy = final_occupancy[ward][0] if isinstance(final_occupancy[
            ward], np.ndarray) else final_occupancy[ward]
        print(f"{ward:<5} {total_admissions[ward]:>12} {total_relocations[
            ward]:>12} {total_losses[ward]:>8} {occupancy:>10} {
            prob_all_beds_occupied[ward]:>10.3f} {expected_admissions[ward
            ]:>15.3f} {expected_relocations[ward]:>17.3f}")

# Display results for each variance
format_results("Results with variance 2/ÎźÂ¸", results_variances_1)
format_results("Results with variance 3/ÎźÂ¸", results_variances_2)
```

```
format_results("Results with variance 4/ÎźÂ¸", results_variances_3)
```

Sensitivity Analysis 2

```
import numpy as np
import pandas as pd
from scipy.stats import lognorm


import random
np.random.seed(42)

# Parameters
wards = ['A', 'B', 'C', 'D', 'E', 'F']
arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
    13.0}
mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20, 'F': 27}
urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
relocation_probs = {
    'A': [0.00, 0.05, 0.10, 0.05, 0.80],
    'B': [0.20, 0.00, 0.50, 0.15, 0.15],
    'C': [0.30, 0.20, 0.00, 0.20, 0.30],
    'D': [0.35, 0.30, 0.05, 0.00, 0.30],
    'E': [0.20, 0.10, 0.60, 0.10, 0.00]
}

def update_relocation_probs(relocation_probs):
    for ward in relocation_probs:
        relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
    relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
    return relocation_probs

def simulate_hospital_with_lognorm(days, adjusted_capacities,
    relocation_probs, variances):
    ward_occupancy = {ward: 0 for ward in adjusted_capacities}
    total_admissions = {ward: 0 for ward in adjusted_capacities}
    total_relocations = {ward: 0 for ward in adjusted_capacities}
    total_losses = {ward: 0 for ward in adjusted_capacities}
    total_occupied_on_arrival = {ward: 0 for ward in adjusted_capacities}

    for day in range(days):
```

```
        for ward in adjusted_capacities:
            arrivals = np.random.poisson(arrival_rates[ward])
            for _ in range(arrivals):
                if ward_occupancy[ward] < adjusted_capacities[ward]:
                    ward_occupancy[ward] += 1
                    total_admissions[ward] += 1
                else:
                    total_occupied_on_arrival[ward] += 1
                    relocated = False
                    for j, prob in enumerate(relocation_probs[ward]):
                        if np.random.rand() < prob:
                            alt_ward = list(adjusted_capacities.keys())[j]
                            if ward_occupancy[alt_ward] <
                                adjusted_capacities[alt_ward]:
                                 ward_occupancy[alt_ward] += 1
                                 total_relocations[alt_ward] += 1
                                 relocated = True
                                 break
                    if not relocated:
                        total_losses[ward] += 1

            mean = np.log(mean_stay[ward]**2 / np.sqrt(variances[ward] +
                mean_stay[ward]**2))
            sigma = np.sqrt(np.log(variances[ward] / mean_stay[ward]**2 + 1)
                )
            departures = np.random.poisson(ward_occupancy[ward] / lognorm.
                rvs(sigma, scale=np.exp(mean), size=1))
            ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)

    prob_all_beds_occupied = {ward: total_occupied_on_arrival[ward] / (
        total_admissions[ward] + total_occupied_on_arrival[ward])
                              for ward in adjusted_capacities}
    expected_admissions = {ward: total_admissions[ward] / days for ward in
        adjusted_capacities}
    expected_relocations = {ward: total_relocations[ward] / days for ward in
         adjusted_capacities}

    return total_admissions, total_relocations, total_losses, ward_occupancy
        , prob_all_beds_occupied, expected_admissions, expected_relocations

def format_results(title, results):
    total_admissions, total_relocations, total_losses, final_occupancy,
        prob_all_beds_occupied, expected_admissions, expected_relocations =
        results
```

```python

    print(f"\n{title}\n")
    print(f"{'Ward':<5} {'Admissions':>12} {'Relocations':>12} {'Losses':>8} {'Occupancy':>10} {'Prob Full':>10} {'Exp Admissions':>15} {'Exp Relocations':>17}")
    print("-" * 90)
    for ward in wards:
        occupancy = final_occupancy[ward][0] if isinstance(final_occupancy[ward], np.ndarray) else final_occupancy[ward]
        print(f"{ward:<5} {total_admissions[ward]:>12} {total_relocations[ward]:>12} {total_losses[ward]:>8} {occupancy:>10} {prob_all_beds_occupied[ward]:>10.3f} {expected_admissions[ward]:>15.3f} {expected_relocations[ward]:>17.3f}")

# Different bed distribution scenarios
# Scenario 1: Increase beds in high-urgency wards (e.g., Ward D)
capacities_scenario1 = initial_capacities.copy()
capacities_scenario1['D'] += 10  # Increase beds in Ward D by 10
capacities_scenario1['A'] -= 3
capacities_scenario1['B'] -= 3
capacities_scenario1['C'] -= 2
capacities_scenario1['E'] -= 2
capacities_scenario1['F'] = 27  # Keep Ward F constant

# Scenario 2: Even distribution of beds
total_beds = sum(initial_capacities.values())
even_distribution = total_beds // len(wards)
capacities_scenario2 = {ward: even_distribution for ward in wards}

# Scenario 3: Increase beds in high-arrival wards (e.g., Wards A and F)
capacities_scenario3 = initial_capacities.copy()
capacities_scenario3['A'] += 10  # Increase beds in Ward A by 10
capacities_scenario3['F'] += 10  # Increase beds in Ward F by 10
capacities_scenario3['B'] -= 5
capacities_scenario3['C'] -= 5
capacities_scenario3['D'] -= 5
capacities_scenario3['E'] -= 5

# Set variances for log-normal distribution
variances = {ward: 2 / (mean_stay[ward] ** 2) for ward in wards}

# Run simulations for each scenario
relocation_probs_updated = update_relocation_probs(relocation_probs.copy())
```

```
results_scenario1 = simulate_hospital_with_lognorm(365, capacities_scenario1
    , relocation_probs_updated, variances)
results_scenario2 = simulate_hospital_with_lognorm(365, capacities_scenario2
    , relocation_probs_updated, variances)
results_scenario3 = simulate_hospital_with_lognorm(365, capacities_scenario3
    , relocation_probs_updated, variances)

# Display results for each scenario
format_results("Scenario 1: Increase beds in high-urgency wards",
    results_scenario1)
format_results("Scenario 2: Even distribution of beds", results_scenario2)
format_results("Scenario 3: Increase beds in high-arrival wards",
    results_scenario3)
```

Sensitivity Analysis 3

```
import numpy as np
import pandas as pd
from scipy.stats import lognorm

# Set random seed for reproducibility
np.random.seed(42)

# Parameters
wards = ['A', 'B', 'C', 'D', 'E', 'F']
arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
    13.0}
mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
relocation_probs = {
    'A': [0.00, 0.05, 0.10, 0.05, 0.80],
    'B': [0.20, 0.00, 0.50, 0.15, 0.15],
    'C': [0.30, 0.20, 0.00, 0.20, 0.30],
    'D': [0.35, 0.30, 0.05, 0.00, 0.30],
    'E': [0.20, 0.10, 0.60, 0.10, 0.00]
}

def update_relocation_probs(relocation_probs):
    for ward in relocation_probs:
        relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
    relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
```

```
    return relocation_probs

def reallocate_beds(total_beds, initial_capacities, urgency_points, bed_capacity_F):
    remaining_beds = total_beds - bed_capacity_F
    sorted_wards = sorted(initial_capacities.keys(), key=lambda x: urgency_points[x], reverse=True)
    adjusted_capacities = {}

    for ward in sorted_wards:
        proportion = initial_capacities[ward] / sum(initial_capacities.values())
        adjusted_capacities[ward] = int(proportion * remaining_beds)

    current_total = sum(adjusted_capacities.values())
    difference = remaining_beds - current_total

    if difference > 0:
        for ward in sorted_wards:
            adjusted_capacities[ward] += 1
            difference -= 1
            if difference == 0:
                break

    adjusted_capacities['F'] = bed_capacity_F
    return adjusted_capacities

def simulate_hospital_with_lognorm(days, adjusted_capacities, relocation_probs, variances):
    ward_occupancy = {ward: 0 for ward in adjusted_capacities}
    total_admissions = {ward: 0 for ward in adjusted_capacities}
    total_relocations = {ward: 0 for ward in adjusted_capacities}
    total_losses = {ward: 0 for ward in adjusted_capacities}
    total_occupied_on_arrival = {ward: 0 for ward in adjusted_capacities}

    for day in range(days):
        for ward in adjusted_capacities:
            arrivals = np.random.poisson(arrival_rates[ward])
            for _ in range(arrivals):
                if ward_occupancy[ward] < adjusted_capacities[ward]:
                    ward_occupancy[ward] += 1
                    total_admissions[ward] += 1
                else:
                    total_occupied_on_arrival[ward] += 1
```

```python
                    relocated = False
                    for j, prob in enumerate(relocation_probs[ward]):
                        if np.random.rand() < prob:
                            alt_ward = list(adjusted_capacities.keys())[j]
                            if ward_occupancy[alt_ward] < adjusted_capacities[alt_ward]:
                                ward_occupancy[alt_ward] += 1
                                total_relocations[alt_ward] += 1
                                relocated = True
                                break
                    if not relocated:
                        total_losses[ward] += 1

            mean = np.log(mean_stay[ward]**2 / np.sqrt(variances[ward] + mean_stay[ward]**2))
            sigma = np.sqrt(np.log(variances[ward] / mean_stay[ward]**2 + 1))
            departures = np.random.poisson(ward_occupancy[ward] / lognorm.rvs(sigma, scale=np.exp(mean), size=1))
            ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)

    prob_all_beds_occupied = {ward: total_occupied_on_arrival[ward] / (total_admissions[ward] + total_occupied_on_arrival[ward])
                              for ward in adjusted_capacities}
    expected_admissions = {ward: total_admissions[ward] / days for ward in adjusted_capacities}
    expected_relocations = {ward: total_relocations[ward] / days for ward in adjusted_capacities}

    return total_admissions, total_relocations, total_losses, ward_occupancy, prob_all_beds_occupied, expected_admissions, expected_relocations

def format_results(title, results):
    total_admissions, total_relocations, total_losses, final_occupancy, prob_all_beds_occupied, expected_admissions, expected_relocations = results

    print(f"\n{title}\n")
    print(f"{'Ward':<5} {'Admissions':>12} {'Relocations':>12} {'Losses':>8} {'Occupancy':>10} {'Prob Full':>10} {'Exp Admissions':>15} {'Exp Relocations':>17}")
    print("-" * 90)
    for ward in wards:
```

```
        occupancy = final_occupancy[ward][0] if isinstance(final_occupancy[
            ward], np.ndarray) else final_occupancy[ward]
        print(f"{ward:<5} {total_admissions[ward]:>12} {total_relocations[
            ward]:>12} {total_losses[ward]:>8} {occupancy:>10} {
            prob_all_beds_occupied[ward]:>10.3f} {expected_admissions[ward
            ]:>15.3f} {expected_relocations[ward]:>17.3f}")

# Define different total bed scenarios
total_beds_scenario1 = 170
total_beds_scenario2 = 180
total_beds_scenario3 = 150

# Set variances for log-normal distribution
variances = {ward: 2 / (mean_stay[ward] ** 2) for ward in wards}

# Calculate optimal bed capacity for Ward F (previously found to be 27)
optimal_bed_capacity_F = 27

# Calculate adjusted capacities for each scenario
adjusted_capacities_scenario1 = reallocate_beds(total_beds_scenario1,
    initial_capacities, urgency_points, optimal_bed_capacity_F)
adjusted_capacities_scenario2 = reallocate_beds(total_beds_scenario2,
    initial_capacities, urgency_points, optimal_bed_capacity_F)
adjusted_capacities_scenario3 = reallocate_beds(total_beds_scenario3,
    initial_capacities, urgency_points, optimal_bed_capacity_F)

# Update relocation probabilities
relocation_probs_updated = update_relocation_probs(relocation_probs.copy())

# Run simulations for each scenario
results_scenario1 = simulate_hospital_with_lognorm(365,
    adjusted_capacities_scenario1, relocation_probs_updated, variances)
results_scenario2 = simulate_hospital_with_lognorm(365,
    adjusted_capacities_scenario2, relocation_probs_updated, variances)
results_scenario3 = simulate_hospital_with_lognorm(365,
    adjusted_capacities_scenario3, relocation_probs_updated, variances)

# Display results for each scenario
format_results("Scenario 1: Total beds = 170", results_scenario1)
format_results("Scenario 2: Total beds = 180", results_scenario2)
format_results("Scenario 3: Total beds = 150", results_scenario3)
```