Computer-Assisted Simulations Using R and RStudio to Assist in Operations Research and Analysis in the Context of Clinical Laboratory Management: A Gentle Introduction and Simple Guide for Pathologists and Laboratory Professionals

Mark Angelo Ang[1] and Karen Cybelle Sotalbo[2]

[1]*Department of Pathology, College of Medicine, University of the Philippines Manila*
[2]*Department of Laboratories, Philippine General Hospital, University of the Philippines Manila*

ABSTRACT

Operations research (OR) is a valuable yet underutilized field in clinical laboratory management, offering practical solutions to optimize workflows, resource allocation, and decision-making. Despite its potential, the adoption of OR methodologies remain limited due to a lack of training and familiarity among pathologists and laboratory professionals. This paper addresses this gap by presenting an accessible introduction and practical guide to analyzing operations research problems in clinical laboratories using computer-assisted simulations in R, implemented within the R Studio environment.

The proposed framework emphasizes simplicity and flexibility, leveraging the extensive capabilities of base R to model and analyze critical OR questions. The paper outlines step-by-step methods for defining problems, constructing simulation models, and interpreting results, ensuring that readers can replicate and adapt these techniques to their unique laboratory contexts.

Key features of the framework include its emphasis on reproducibility, customization, and the integration of data-driven insights into decision-making processes. Case studies and examples drawn from real-world laboratory scenarios illustrate the application of R simulations to address challenges such as minimizing turnaround times, balancing staffing levels, and managing inventory efficiently.

This guide aims to empower laboratory professionals and pathologists with the tools and skills to integrate operations research into their practice, fostering a culture of innovation and efficiency in clinical settings. By bridging the gap between OR theory and practical application, this paper contributes to the broader adoption of computational approaches in laboratory management, ultimately enhancing the quality and sustainability of healthcare services.

Key words: operations research, clinical laboratory management, simulation modeling, R programming, healthcare resource management

ISSN 2507-8364 (Online)
Printed in the Philippines.

Received: 26 November 2024.
Accepted: 9 December 2024.
Published online first: 17 December 2024.
https://doi.org/10.21141/PJP.2024.14

Corresponding author: Mark Angelo C. Ang, MD
E-mail: mcang1@up.edu.ph
ORCiD: https://orcid.org/0000-0003-1292-9493

INTRODUCTION

Operations research (OR) is critical in clinical laboratory management as it provides a structured, data-driven approach to optimizing operations, improving efficiency, and ensuring quality service delivery. In modern laboratories, challenges such as high testing volumes, limited resources, and stringent turnaround time (TAT) requirements necessitate robust analytical tools. OR methods such as queuing theory, simulation modeling, and optimization algorithms enable laboratories to identify bottlenecks, optimize resource allocation, and enhance workflow efficiency.[1-3] For example, discrete-event simulation has been used to reduce sample processing delays, improving patient outcomes while minimizing costs.[4] These techniques help laboratories adapt to demand fluctuations, especially during pandemics or seasonal surges, ensuring they remain agile and resilient.

Beyond operational efficiency, OR supports strategic decision-making by forecasting future testing demands,

determining cost-effective inventory policies, and planning laboratory expansions. This is crucial in an era of precision medicine, where diagnostic labs play a pivotal role in healthcare. For instance, predictive analytics informed by OR can help prevent reagent stockouts, avoiding costly delays in diagnostic processes.[5-7] Furthermore, OR enhances the ability to meet accreditation and regulatory requirements by ensuring processes are both efficient and compliant.

Operations research bridges the gap between operational efficiency and strategic foresight, making it indispensable for managing clinical laboratories in today's complex healthcare ecosystem.

Operations research and analysis is an approach to answer questions that arise in the context of clinical laboratory management and focus on efficiency and optimization problems.

Operations research in clinical laboratories is the application of analytical methods to optimize the use of resources, such as staff, equipment, and reagents, to improve efficiency and reduce costs without compromising the quality of diagnostic services.[8] OR involves the use of mathematical modeling, simulation, and statistical analysis to support decision-making processes in laboratory management, such as workload balancing, test prioritization, and process redesign.[2,9,10] In the context of clinical laboratories, operations research focuses on improving workflow efficiency, reducing turnaround times, and ensuring timely delivery of test results to meet patient care demands.[11] OR applies quantitative techniques to manage laboratory quality, predict future testing demands, and design scalable operations to accommodate growth while maintaining high standards of service.

Important questions arise in the conduct of operations management of the clinical laboratory.

Workflow optimization

Workflow optimization issues in clinical laboratories arise in various settings, including high-volume testing centers, specialized labs, STAT and emergency testing areas, and facilities responding to public health crises. Challenges often include bottlenecks in sample processing, resource allocation inefficiencies, and disruptions from urgent test prioritization or sudden demand surges. Small laboratories with limited resources and those transitioning to new technologies also face delays due to constrained capacity or misaligned workflows.[12,13] Addressing these challenges requires tailored strategies, such as leveraging automation, improving resource planning, and implementing dynamic queuing systems. Examples of questions that may arise are:

- *How can we reduce the turnaround time (TAT) for routine and urgent test results?*
- *What is the optimal sequence for processing different types of specimens (e.g., blood, urine, tissue)?*
- *Where are the bottlenecks in the laboratory workflow, and how can they be alleviated?*
- *How can we ensure that critical tests (e.g., STAT tests) are prioritized without disrupting routine workflows?*

Resource allocation

Resource allocation issues in clinical laboratories commonly arise in settings with high variability in demand, such as during peak testing hours in high-volume labs or public health emergencies. Limited staffing, budget constraints, and equipment availability exacerbate these challenges, particularly in rural or small-scale laboratories with fewer resources.[14] STAT and emergency testing areas frequently face resource allocation conflicts, as prioritizing urgent tests can disrupt routine workflows and strain personnel and equipment.[15,16] Additionally, laboratories transitioning to automation or expanding services often encounter temporary inefficiencies as resources are diverted to implement new systems or train staff.[8] These settings highlight the need for optimized resource allocation strategies to balance demand, costs, and service quality. Examples of questions that may arise include:

- *What is the optimal number of staff required for peak and off-peak hours?*
- *How should staff be scheduled to minimize overtime and maximize efficiency?*
- *How can we maximize the utilization of high-cost equipment (e.g., hematology analyzers, mass spectrometers)?*
- *What is the optimal maintenance schedule to minimize downtime?*

Inventory and supply chain management

Issues in inventory and supply chain management commonly arise in clinical laboratories with fluctuating demand, such as during seasonal surges or public health emergencies like pandemics. Laboratories often face challenges in forecasting reagent and consumable usage, leading to overstocking, stockouts, or waste, particularly in high-volume testing facilities.[17] Small or rural laboratories, operating with limited budgets, may encounter difficulties in maintaining optimal inventory levels due to constrained purchasing power and delayed supplier deliveries.[18] Additionally, disruptions in global supply chains, such as those experienced during COVID-19, can exacerbate shortages, affecting both routine and emergency testing capabilities.[19] These settings emphasize the importance of implementing robust inventory management systems and dynamic supply chain strategies to ensure reliable and cost-effective operations. Example of questions that may arise include:

- *What are the ideal inventory levels for reagents to prevent stockouts while minimizing holding costs?*
- *How can we forecast demand for reagents based on historical testing patterns?*
- *What is the most cost-effective way to manage procurement and logistics for laboratory supplies?*

Quality and accuracy

Issues in quality and accuracy in clinical laboratories often arise in settings with high workloads, complex testing protocols, or inadequate quality control measures. Laboratories handling large volumes of routine or specialized tests may encounter errors during pre-analytical, analytical, or post-analytical phases due to rushed procedures or insufficient staff training.[20] Resource-limited laboratories, such as those in rural or underfunded healthcare systems, often face challenges in maintaining consistent quality due to outdated equipment, lack of standard operating procedures, or insufficient quality control practices.[21] Additionally, emergency testing environments or laboratories responding to pandemics may experience a higher risk of errors due to the pressure

to deliver rapid results without compromising accuracy.[22] These settings highlight the critical need for robust quality management systems and ongoing staff education to ensure diagnostic reliability. Examples of questions that may arise are:

- *How can we minimize pre-analytical, analytical, and post-analytical errors?*
- *What is the impact of process changes on the rate of quality control failures?*
- *What is the optimal frequency for running quality control samples to balance cost and error detection?*

Capacity planning

Issues in capacity planning in clinical laboratories arise in settings experiencing unpredictable demand, such as during seasonal epidemics or public health crises like pandemics. Laboratories often struggle to scale resources, equipment, and staffing to meet sudden surges, resulting in delayed turnaround times and service disruptions.[23,24] Additionally, specialized laboratories offering high-complexity tests often grapple with capacity constraints due to the limited availability of skilled personnel and high-cost equipment.[25] These challenges underscore the importance of data-driven forecasting and dynamic resource allocation to optimize laboratory capacity and responsiveness. Example of specific questions that may arise include:

- *How should the laboratory plan for fluctuations in test volumes (e.g., seasonal trends, pandemics)?*
- *What infrastructure investments are needed to handle projected growth in testing demand?*
- *How can laboratory layout be optimized to improve workflow and accommodate growth?*

Cost management

Issues in cost management in clinical laboratories arise in settings where balancing quality and efficiency is critical, particularly in resource-limited environments such as rural or small-scale labs. High operational costs, driven by reagents, equipment maintenance, and staff salaries, often strain budgets, especially when reimbursement rates do not align with testing costs.[26] Large laboratories with high test volumes may face inefficiencies due to overuse or waste of consumables, while smaller labs may struggle with the economies of scale needed to reduce per-test costs.[26] Public health laboratories or those responding to crises often experience cost escalations due to sudden surges in testing demand, necessitating emergency procurement and overtime staffing.[27] These challenges highlight the importance of implementing cost-tracking systems, optimizing resource utilization, and aligning financial strategies with operational goals to ensure sustainability.

- *What is the cost per test for different assays, and how can it be reduced without compromising quality?*
- *How should pricing models be adjusted to remain competitive while ensuring profitability?*
- *How can the laboratory allocate its budget to maximize operational efficiency and quality?*

Patient care perspective type of questions

Issues in patient care arise in clinical laboratories when delays, errors, or inefficiencies compromise the timely delivery of accurate test results, which are critical for clinical decision-making. High-volume laboratories may face bottlenecks or prolonged turnaround times (TAT), particularly during peak testing periods or public health emergencies, leading to delays in treatment initiation.[11,28] Resource-limited or rural laboratories often struggle to maintain consistent quality due to outdated equipment or insufficient staff, increasing the likelihood of diagnostic errors.[29] Laboratories managing STAT and emergency testing may also encounter challenges in prioritizing critical samples without disrupting routine workflows, potentially impacting patient outcomes.[30] These issues highlight the importance of efficient workflows, robust quality control, and reliable communication with healthcare providers to ensure optimal patient care.

- *What is the optimal TAT for different test categories to meet clinical needs?*
- *How can patient satisfaction be improved through better communication of test results?*

Popular methods exist that answer operations research questions in the context of clinical laboratory management

To address operations research (OR) challenges in clinical laboratories, a variety of methods and techniques are employed, each tailored to specific problems. Mathematical optimization techniques, such as Linear Programming (LP), Integer Programming (IP), and Mixed-Integer Linear Programming (MILP), are foundational tools for resource allocation and scheduling.[31,32]

LP effectively allocates resources, including reagents and staff, to minimize costs, while IP focuses on discrete decisions, such as determining the optimal number of instruments or staff shifts. MILP bridges continuous and discrete variables, making it suitable for complex tasks like laboratory expansions or integrating new technologies.[31,32] Despite their precision and scalability, these methods often demand detailed data and advanced expertise, presenting challenges for smaller or resource-constrained laboratories.[33]

Simulation modeling and queuing theory are particularly effective for addressing the dynamic and stochastic nature of laboratory workflows. Discrete Event Simulation (DES) models workflows to identify bottlenecks and evaluate the impact of process changes, while Monte Carlo simulations manage uncertainty by modeling variability in sample arrival rates or test demand.[34,35]

Queuing theory complements these methods by analyzing sample flow and optimizing service capacity, particularly in high-priority settings like STAT labs, where rapid processing is critical.[36,37] Although these approaches offer a risk-free environment for testing scenarios, they require significant time for model development and depend on high-quality data inputs, limiting their feasibility in certain settings.

These approaches, though effective, often require specialized expertise and tools, which can hinder their implementation in laboratories lacking dedicated OR personnel.

Collectively, these methodologies form a robust toolkit for addressing the multifaceted challenges in clinical laboratory management. While each method has its unique strengths and limitations, their strategic application—alone

or in combination—can significantly enhance efficiency, reduce costs, and support data-driven decision-making. By tailoring these tools to specific laboratory needs and investing in the necessary expertise, laboratories can overcome operational hurdles and deliver reliable, high-quality diagnostic services.

Use of computer assisted simulations

Computer-assisted simulations have emerged as a transformative tool for analyzing complex systems, enabling the study of dynamic processes without real-world disruptions. The roots of simulation as a method can be traced back to the 1940s, when the Monte Carlo method was developed during the Manhattan Project to model neutron diffusion in nuclear reactions.[38,39] In the decades that followed, advancements in computing technology significantly enhanced simulation capabilities, leading to the development of discrete-event simulation (DES) in the 1950s and 60s, which became a cornerstone for studying queuing systems and logistics. Early adopters in industries like manufacturing and defense found these methods invaluable for optimizing resource allocation and process efficiency.[34]

The rise in popularity of computer-assisted simulations has been driven by improvements in computational power, accessibility of software, and the growing complexity of systems requiring analysis. Modern simulation tools, including those for system dynamics, agent-based modeling, and hybrid approaches, are now widely used in healthcare, logistics, and engineering. Open-source programming languages like R and Python have democratized access to simulation tools, enabling researchers and practitioners to model real-world problems without the need for expensive proprietary software.[40,41] In clinical laboratory management, simulations have become essential for addressing challenges such as high testing volumes, resource constraints, and demand fluctuations, further cementing their role as an indispensable decision-making tool.[42,43]

Theory behind computer assisted simulations: a brief conceptual description

Computer-assisted simulations in operations research (OR) are grounded in mathematical modeling and probability theory. At their core, simulations replicate the behavior of complex systems by iteratively calculating outcomes based on predefined mathematical rules and probabilistic distributions.[44] DES, one of the most widely used methodologies, models systems as a sequence of events that occur at discrete points in time. These events are governed by probabilistic distributions such as exponential or Poisson, which describe stochastic processes like inter-arrival times or service durations. Monte Carlo simulations, another key technique, use random sampling to solve deterministic or probabilistic problems by exploring a range of possible outcomes under varying assumptions.[38] Both approaches rely on random number generation and iterative computation to model uncertainty and variability, which are central to real-world OR problems.

The mathematical foundation of simulations also incorporates optimization and queuing theory to analyze system performance. For example, queuing models are built using Markov chains and probability distributions to estimate metrics such as average wait times, service utilization, and system capacity.[45] Optimization models, often integrated into simulation frameworks, use linear or nonlinear programming to identify optimal resource allocation strategies. Simulation algorithms are designed to mimic real-world processes iteratively, capturing dynamic interactions among variables while accounting for randomness. This capability makes simulations particularly powerful for modeling complex, interdependent systems like clinical laboratories, where exact analytical solutions may not exist due to high variability and uncertainty. By combining mathematical rigor with computational efficiency, simulations provide actionable insights for OR practitioners.[45]

Popular software used in computer assisted simulations

Operations research employs a variety of software tools to address complex decision-making problems across different domains. Among the most popular software used in OR are optimization and simulation tools. Optimization software such as Solver, POM-QM, and Lingo are frequently utilized in educational settings to enhance students' problem-solving capabilities, as evidenced by their effective use in a public university in Lima, Peru, where they significantly improved student performance in OR courses.[46]

Additionally, Maple is highlighted for its optimization features, which are particularly beneficial in educational contexts for formulating, solving, and visualizing optimization models.[47]

Simulation software also plays a crucial role in OR, with tools like VISSIM, TSIS/CORSIM, and SYNHRO/SIMTRAFFIC being extensively used for traffic operations, each offering unique capabilities for modeling various traffic conditions and providing measures of effectiveness.[48] The widespread use of simulation modeling in fields such as manufacturing, health, and military applications has led to a proliferation of simulation software tools, as noted in a survey by the Operational Research Society of Great Britain.[49]

Furthermore, the development of micro-computer simulation languages has made powerful and inexpensive simulation tools accessible, enhancing the capabilities of OR analysts.[50] The integration of advanced software tools in OR and analytics, such as machine learning in R and algebraic modeling with JuMP, is increasingly important for handling large data sets and complex models, as demonstrated in a course designed to equip students with these skills.[51]

Overall, the diverse range of software tools available in OR, from optimization to simulation, underscores their critical role in facilitating effective decision-making and problem-solving across various industries and educational settings.

Challenges in operations research

Current challenges in applying operations research (OR) to clinical laboratory management stem from both technical and organizational factors. One major issue is the complexity of laboratory workflows, which involve multiple interconnected processes such as sample collection, processing, quality control, and reporting. Modeling

these workflows requires advanced analytical methods like simulation modeling or optimization techniques, which are often beyond the expertise of laboratory personnel.[52] Additionally, clinical laboratories face resource constraints, such as limited access to specialized software and computational tools, further complicating the integration of OR into routine decision-making. High variability in demand, driven by external factors like pandemics, adds another layer of complexity, requiring dynamic models that are challenging to design and implement effectively.[53]

Another significant challenge is the widespread lack of quantitative and analytical skills among laboratory managers and staff. Many professionals in clinical laboratory management are not trained in the rigorous methodologies required for consistent and reproducible OR studies.[54] As a result, even when data are available, there is a risk of oversimplified or inconsistent approaches that fail to capture the nuances of real-world laboratory operations. Moreover, inadequate training in statistical programming languages (e.g., R or Python) and software for simulation and optimization limits the ability to adopt a reproducible framework. This gap in skills leads to missed opportunities for leveraging OR techniques to improve efficiency, reduce costs, and enhance patient care. Bridging this gap requires targeted training programs, cross-disciplinary collaboration, and investment in user-friendly OR tools tailored to the clinical laboratory setting.[55]

Despite challenges, operations research and analysis can be done using accessible open-source software like R and implemented in RStudio.

Operations research (OR) and analysis can be effectively conducted using accessible open-source tools like R in RStudio, provided users follow key simple practices.

R is a powerful statistical programming environment that offers extensive libraries for simulation, optimization, and statistical modeling, making it highly suitable for a wide range of operations research (OR) applications, including resource allocation and workflow optimization. The language is renowned for its flexibility and extensibility, allowing users to perform a variety of statistical analyses such as regression, ANOVA, time series, and multivariate analysis, which are crucial for OR tasks.[56] R's open-source nature and the availability of over 4000 specialized packages further enhance its capabilities, providing users with tools for data manipulation, calculation, and graphical display. The language's ability to handle complex data structures and perform advanced statistical modeling is particularly beneficial in OR, where data-driven decision-making is essential.[56] Moreover, R's algebraic modeling approach, as discussed in optimization contexts, allows for clear formulation and implementation of linear and mixed-integer optimization problems, which are common in OR. The integration of R with data analytics platforms reduces the barrier to entry for professionals, enabling them to leverage data analytics tools effectively.[57]

Furthermore, R's open-source nature and vast community support reduce barriers to entry, allowing clinical laboratory managers to adopt OR techniques without expensive software. By combining these practices with RStudio's user-friendly interface, even individuals with basic programming skills can harness the power of OR to make data-driven decisions in laboratory management.

Computer-assisted simulations in R using RStudio are highly effective and versatile due to the power of the R programming language and the integrated development environment (IDE) provided by RStudio.

Computer-assisted simulations in R using RStudio are highly effective and versatile due to the robust capabilities of the R programming language and the integrated development environment provided by RStudio. R is a preferred choice for simulation studies across various fields due to its extensive library of packages and the ability to write custom functions, which is particularly beneficial in areas like computer adaptive testing (CAT) and item response theory (IRT) simulations.[58]

The RStudio IDE enhances this experience by offering a user-friendly interface that facilitates project management, script writing, and access to comprehensive documentation, making it an ideal environment for empirical research and educational purposes.[59]

In educational settings, R is utilized to teach complex statistical concepts through simulations, such as sampling distributions and hypothesis testing, which help students grasp abstract ideas more concretely.[60] Additionally, R's capabilities extend to large-scale simulation studies, where packages like *simsalapar* enable efficient parallel computing and comprehensive result analysis, crucial for handling complex quantitative risk management problems.[61]

The RxODE package further exemplifies R's versatility by allowing the simulation of differential equation models, which can be integrated into interactive applications for real-time scenario evaluation.[62]

By combining R's statistical prowess with RStudio's integrated environment, users can create flexible, reproducible, and cost-effective simulations to address complex operations research questions in various fields, including clinical laboratory management. This powerful combination democratizes advanced analytics, making it accessible to a wide range of users and applications.

Simulations can be performed using base R functions to answer simple OR questions using loops.

Even if access to custom packages specifically designed for operations research questions are not available, simulations for simple operations research (OR) questions can be effectively performed using base R functions, leveraging its built-in capabilities such as loops (for, while repeat) and vectorized operations.

This approach provides a flexible and straightforward framework for tackling basic OR problems without requiring specialized packages. Base R excels in simulation tasks through its powerful constructs: vectorized operations enable efficient computations over large datasets, while control structures like loops and conditional statements (if, else) support iterative and dynamic process modeling.

Additionally, R's random number generation functions (runif, rnorm, rpois, and sample) facilitate stochastic

modeling, making it ideal for tasks such as modeling queue behavior, conducting Monte Carlo simulations, or analyzing inventory dynamics. These features collectively ensure that base R is well-suited for addressing a range of simple OR challenges.

Step-by-step instructions for installing R and RStudio

To perform simulations using R, you need to download and install the software on a PC or Mac with an active internet connection. The following text is a step-by-step guide to installing and setting up R and R studio.

Step 1: Download R

1. Visit the CRAN (Comprehensive R Archive Network).
2. Choose your operating system:
   - Windows: Click on "Download R for Windows," then select "base" and download the installer.
   - macOS: Click on "Download R for macOS" and choose the appropriate file for your macOS version.
   - Linux: Follow the detailed instructions for your distribution (Ubuntu, Fedora, etc.).
3. Run the installer:
   - Follow the prompts, accept the defaults, and complete the installation.

Step 2: Download RStudio

1. Visit the RStudio Download Page.
2. Select the free version *(RStudio Desktop – Open Source License)* and download the installer for your operating system.
3. Run the installer:
   - Follow the instructions and complete the installation.

Step 3: Verify installation

1. Open **RStudio**.
2. Verify that R is linked to RStudio by typing the following command in the console:

```
================================
Version
================================
```

The output should display the installed R version and platform information.

Good practices when running R in RStudio

1. Organize your workspace

A well-organized workspace is critical for efficient coding in RStudio. Begin by setting a working directory to define the location where your files will be saved and accessed. This can be done using the *setwd()* function or by navigating to *Session > Set Working Directory* in RStudio's menu. For example:

```
================================
setwd("C:/Your/Directory/Path")
================================
```

Note: replace "*C:/Your/Directory/Path*" with the specific path on your computer for your project.

Additionally, keep projects separate by using RStudio's Projects feature, which allows you to create isolated environments for different analyses. This helps maintain a clean and focused workflow, especially when managing multiple tasks simultaneously.

2. Comment your code

Commenting your code is essential for clarity and collaboration. Use comments to explain the purpose of each section, making your script easier to understand for yourself and others in the future. For instance:

```
================================
# Generate 100 random numbers from a normal distribution
random_numbers <- rnorm(100, mean = 0, sd = 1)
================================
```

Comments have the symbol "#" at the start. This practice ensures that your logic and intent remain clear and human readable, even when revisiting code after a long period or sharing it with team members.

3. Use version control

Integrating version control, such as Git, within RStudio is invaluable for tracking changes in your code. Version control allows you to maintain a history of edits, collaborate with others, and revert to previous versions when needed. RStudio's Git integration makes it easy to commit changes, push updates, and manage branches, ensuring a robust development workflow.

4. Save your work

Always save your scripts regularly to avoid losing progress. Use .R files to organize your code and the Source Pane in RStudio to write, save, and run scripts systematically. Working from the Source Pane instead of directly in the console provides better control and allows you to rerun sections of your code easily.

5. Utilize packages wisely

Efficient package management is crucial for a streamlined workflow. Install only the packages you need using:

```
================================
install.packages("ggplot2")
================================
```

At the beginning of your script, load these packages explicitly with library() to ensure they are available for use:

```
================================
library(ggplot2)
================================
```

This practice keeps your environment organized and avoids potential conflicts or redundancies. Note that functions overwrite previously loaded ones if they have the same name.

6. Keep R and RStudio updated

Regular updates to R, RStudio, and installed packages ensure compatibility, improved functionality, and enhanced security.

The version of R used in the preparation of this manuscript is R version 4.4.1 (2024-06-14 ucrt) -- "Race for Your Life" Copyright © 2024 The R Foundation for Statistical Computing. Platform: x86_64-w64-mingw32/x64.[63] The version of RStudio used in this paper is RStudio 2024.09.0 Build 375 "Cranberry Hibiscus" Release (c8fc7aee, 2024-09-16) for Windows.[64] To update all packages at once, use:

```
==================================
update.packages()
==================================
```

Keeping your software up-to-date prevents bugs and ensures access to the latest features and improvements. This can also be done by pointing and clicking.

7. Write reproducible code

Reproducibility is a hallmark of good programming. For simulations results to be reproducible, a random seed must be set:

```
==================================
set.seed(123)
==================================
```

Additionally, avoid hard-coding file paths by using relative paths, which make your code portable across different systems and environments. A relative path assumes that the starting point of the path is the current working directory and will look for the file of interest beginning in the current working directory.

8. Debug effectively

Debugging is an essential part of coding. Use diagnostic functions like *summary()*, *str()*, and *print()* to inspect variables and identify issues. Break your code into smaller sections and test each one before running the entire script. This incremental approach helps isolate errors quickly and ensures smooth execution.

9. Utilize RStudio features

RStudio offers several features to enhance productivity:

- **Code completion**: Autocomplete saves time and reduces typos.
- **Plots pane**: View visualizations directly within RStudio without cluttering your workspace.
- **Environment pane**: Monitor active variables, their data types, and values to keep track of your workspace.

10. Save outputs

Exporting results, such as plots or data, is straightforward in RStudio. For example, save a plot as a PNG file using:

```
==================================
png("plot.png")
plot(rnorm(100))
dev.off()
==================================
```

This ensures your outputs are preserved and accessible for reporting or further analysis.

Maintain an efficient workflow

To optimize your workflow, use shortcuts like Ctrl + Enter (Windows/Linux) or Cmd + Enter (macOS) to quickly run lines of code. Regularly clear your environment using: rm(list = ls()) to avoid memory issues caused by unnecessary variables. Finally, back up your scripts and results in version-controlled or cloud-based environments for security and easy retrieval.

By following these practices, you can create a structured, efficient, and reproducible environment for coding in RStudio, enhancing your productivity and ensuring the quality of your work.

Key steps in performing a computer-assisted simulation study using base R to answer common operations research (OR) questions:

Once you have installed and set up R and Rstudio, you should be ready to perform simulation experiments and analyses to support operations research. The following paragraphs outline a step-by-step guide to perform simulations in R.

Step 1: Clarify the research or study question

Define the problem

Clearly state the objective of the study. Common problems that arise in the setting of laboratory management include optimizing laboratory workflow, minimizing reagent stockouts, or determining optimal staffing levels.

Example: *"What are the chances of a reagent stockout given fluctuating demand and a fixed supply?"*

Specify the outcome

Identify the metric or outcome you wish to measure. Common outcomes of interest for a clinical laboratory include waiting time, cost, or stockout probability.

Step 2: Define variables and assumptions

List input variables

Identify the key inputs of interest relevant to the operations problem, for example, inter-arrival times, service rates, and demand distribution. This part requires prior information about the specific experience of the clinical laboratory. The researcher or manager must make sure that the assumed parameters closely approximate the distributional characteristics of the phenomena under consideration.

Example: *Daily reagent demand follows a normal probability distribution with a mean of 500 units and standard deviation of 50.*

Specify constraints and assumptions

Document any constraints or fixed parameters (e.g., available supply, service capacity). These are factors that usually have the most impact on the decision-making process and pertain to limits on resources.

Example: *Daily reagent supply is fixed at 600 units.*

Step 3: Set up the model

Choose a simulation approach
Decide on the type of simulation (e.g., discrete-event simulation, Monte Carlo simulation).
Example: *Monte Carlo simulation to estimate the probability of stockouts.*

Define logical flow
Create a logical sequence of events (e.g., sample arrival → service start → service completion).

Plan iterations
Decide on the number of iterations to ensure statistical reliability.

Step 4: Draft the code in base R

Initialize parameters
Define all fixed parameters (e.g., supply level, number of iterations).

```
==========
set.seed(123) # Ensure reproducibility of results
simulations <- 1000
supply <- 600
mean_demand <- 500
sd_demand <- 50
==========
```

Generate random data
Use base R functions to model stochastic inputs. This step is the key step in simulation studies as choice of the stochastic (probabilistic) model has direct effect on the results. The stochastic (probabilistic) model should closely mimic the natural evolution of the phenomenon under consideration as much as practically necessary.

```
==========
demand <- rnorm(simulations, mean = mean_demand, sd = sd_demand)
==========
```

Simulate the process
Use loops and conditional logic to simulate the system. The loop section instructs the computer to perform the simulations many times to adequately model the uncertainty of the phenomena.

```
==========
stockouts <- 0

for (i in 1:simulations)  {
  if (demand[i] > supply)  {
    stockouts <- stockouts + 1
  }
}
==========
```

Calculate the outcome
Compute the desired metric or outcome.

```
==========
probability_stockout <- stockouts / simulations
==========
```

Step 5: Run and debug the code

Run incrementally
Run the code block by block to identify potential errors. This method allows the researcher to fine check the code for errors.

Validate logical flow
Print intermediate results (e.g., head(demand)) to ensure inputs and outputs are logical.

Step 6: Interpret results

Analyze the outputs
Examine the computed outcomes in relation to the research question.

```
==========
print(probability_stockout) # Probability of stockout
==========
```

Provide insights
Example Interpretation: *"The simulation estimates a 4.4% probability of stockout. Based on the current operational situation of the clinical laboratory, this result indicates a potentially preventable loss of productivity and is an opportunity to institute pre-emptive action by increasing the supply or buffer stock."*

Step 7: Ensure reproducibility

Set a random seed
Use set.seed() to ensure consistent results in repeated runs. The computer only generates pseudorandom numbers. Setting the seed at the beginning of the code allows others to faithfully reproduce the results even if the code involves several iterations of generating "random" runs or samples from a probability distribution or stochastic model. Any number can be used, but it must be declared to allow reproducibility.

Example: set.seed(123) #setting the seed to 123 is a simple default strategy, but it can be any number.

Document code
Comment code extensively to explain each step. This allows others to understand the flow and purpose of the code. All characters in a line of code after the pound "#" sign is interpreted by R as a comment. Every next line is considered a new instruction by R.

Example:

```
==========
# Generate daily demand using normal distribution
demand <- rnorm(simulations, mean = mean_demand, sd = sd_demand)
==========
```

Export results
Save results for future reference.

```
==========
write.csv(data.frame(demand, supply), "simulation_results.csv", row.names = FALSE)
==========
```

Step 8: Validate and refine

Test for robustness
Run the simulation with different parameters (e.g., supply levels) to test sensitivity.

Compare results
Validate outputs against known benchmarks or expert estimates.

Step 9: Report findings

Summarize key results
Present the simulation results in clear, concise terms.
Example: *"The Monte Carlo simulation revealed a stockout probability of 4.4%, suggesting a moderate risk under current supply conditions."*

Visualize results
Although there are many excellent packages in R for visualization, the base R plotting functions for visualization are already powerful for basic applications. The following is code that generates a histogram of daily demand:

```
==========
hist(demand, main = "Distribution of Daily Demand", xlab = "Daily Demand")
==========
```

Recommend actions
Translate findings into actionable recommendations. This is one of the most important parts of the analysis as it clearly communicates the findings to decision makers.

This step-by-step process ensures clarity, reproducibility, and actionable outcomes for any OR simulation study conducted in base R.

Illustrative case examples using Base R to demonstrate the analysis of simple OR questions

In the following paragraphs, using typical case scenarios, we will walk you through the process of using R to perform operations research and analysis.

A. Patient or sample queue simulation

Vignette
A typical small government clinical laboratory receives a steady stream of patient samples for processing. Due to the limited resources that the laboratory is given, it is unsurprising that there are days when samples pile up and wait to be processed. You, as the laboratory manager wants to estimate the waiting times to identify bottlenecks in the workflow. You intend to use this information to justify increase in budget allocation for the laboratory.

Strategy
Simulate the waiting times of patients (i.e. samples) in your laboratory where there is only one receiving and processing staff. Assume that the service time follows an exponential distribution with rate = 1.5. Run the simulation for 10 samples.

Solution using base R

```
==========
set.seed(123)
arrival_times <- cumsum(rexp(10, rate = 1)) # Inter-arrival times
service_times <- rexp(10, rate = 1.5) # Service times

waiting_times <- numeric(length(arrival_times))
end_service <- 0 # End time of the previous service

for (i in seq_along(arrival_times))  {
  start_service <- max(arrival_times[i], end_service)
  end_service <- start_service + service_times[i]
  waiting_times[i] <- start_service - arrival_times[i]
}
waiting_times
==========
```

Expected output (waiting time in hours)
[1] 0.00000000 0.09327643 0.00000000 0.15576506 0.35096597
[6] 0.15998745 0.41228424 1.30915313 0.00000000 0.36480311

```
==========
```

Interpretation
In a simulation of arrival times and processing times of the next 10 patients/samples, three (3) patients/samples experience no wait time (0 hours), while the rest of the patients/samples experience waiting times ranging from 6 minutes to 79 minutes.

Explanation and notes
This R code simulates a queuing system where patients/samples arrive and get processed, calculating the waiting time for each patient/sample based on their arrival and service times. The random seed is set using `set.seed(123)` to ensure reproducibility, so the generated random numbers remain consistent across runs.

The arrival times of 10 patients (or samples) are simulated using `cumsum(rexp(10, rate = 1))`, where `rexp(10, rate = 1)` generates random inter-arrival times from an exponential distribution with lambda of 1 (1 patient arrives per hour on average), and `cumsum` calculates their cumulative sum to determine actual arrival times.

Similarly, service (or processing) times are simulated using `rexp(10, rate = 1.5)`, which generates random service durations from an exponential distribution with a rate of 1.5 (1.5 patient samples are processed per hour on average).

A numeric vector, `waiting_times`, is initialized to store the waiting times for each patient (or sample), starting with zeros. The variable `end_service` is set to 0 to track when the previous patient's (or sample's) processing time ends.

The `for` loop iterates over each customer, calculating their waiting time.

For each patient (or sample), the processing time starts either at their arrival time or when the previous customer's service ends, whichever is later.

This is computed using `start_service <- max(arrival_times[i], end_service)`. The service end time is updated as the sum of the processing start time and the current patient's (or sample's) processing time. The waiting time for each patient (or sample) is calculated as the difference between their processing start time and arrival time and is stored in the `waiting_times` vector.

At the end of the simulation, the `waiting_times` vector contains the waiting times for all 10 patients. This code models a first-come, first-served queuing system using an exponential distribution for arrival and processing times, a common approach in queuing theory to represent random events such as patient arrivals and processing durations.

B. Monte Carlo simulation

Vignette

A laboratory faces uncertainty in reagent consumption due to fluctuating daily test demands for a cartridge-based PCR assay for an infectious disease. The lab manager uses Monte Carlo simulation to estimate the probability of running out of reagents (stockouts) when supply is fixed at 600 units per day.

Strategy

Estimate the probability of a reagent stockout. Assume that based on the lab's previous 3 months census, the demand follows a normal probability distribution with mean demand of 500 and an SD of 50. Assume that there is no strong reason to believe that the probability distribution of the demand for the next several months shall be significantly different from the previous 3 months.

Solution using base R

```
==================================
set.seed(123)
simulations <- 1000
demand <- rnorm(simulations, mean = 500, sd = 50) # Daily demand
supply <- 600 # Fixed supply

stockouts <- 0

for (i in 1:simulations)  {
  if (demand[i] > supply)  {
    stockouts <- stockouts + 1
  }
}

probability_stockout <- stockouts / simulations
probability_stockout
==================================

Expected output
[1] 0.028
==================================
```

Interpretation

The probability of experiencing a stockout is approximately 2.8%. Depending on the laboratory's operational strategy, this information may represent an unacceptable level of risk (potential income loss), and thus presents as an opportunity to adjust buffer stock levels to reduce the operational risk brought about by insufficient stocks. Provided that the assumptions are reasonably accurate, this information helps the lab manager decide whether to increase the buffer stock levels or not and how much to increase.

Explanation and notes

This R code simulates a scenario to estimate the probability of a stockout, which occurs when customer demand exceeds available supply. The simulation runs 1,000 iterations to approximate this probability under given demand and supply conditions.

The random seed is set using set.seed(123) to ensure that the generated random numbers are consistent and reproducible. The daily demand is modeled as a normal distribution using *rnorm(simulations, mean = 500, sd = 50)*, which generates 1,000 random values with a mean of 500 and a standard deviation of 50. This represents the variability in daily demand. The supply is fixed at 600 units, indicating the maximum quantity available each day.

A variable stockout is initialized at 0 to count the number of times demand exceeds supply during the simulations. A for loop iterates through each simulated day, comparing the daily demand (demand[i]) to the fixed supply. If the demand on a particular day exceeds the supply, the stockouts counter is incremented by 1.

After completing the loop, the probability of a stockout is calculated as the ratio of the number of stockouts to the total number of simulations (probability_stockout <- stockouts / simulations). This probability provides an estimate of how often demand will surpass supply, helping to assess the risk of insufficient inventory under the given conditions.

Finally, the value of probability_stockout is returned, representing the likelihood of experiencing a stockout based on the simulated data.

C. Inventory Management

Vignette

A clinical laboratory needs to optimize its reagent order quantity for a particular type of PCR cartridges to minimize total costs, which include ordering costs and holding costs. Using a simple inventory model, the lab manager was tasked to determine the optimal order size.

Strategy

Simulate and find the optimum order quantity size using a simple inventory model. Assume that the order cost is 50 (in thousands of pesos) per order, the annual holding cost per unit is 2 (in thousands of pesos), and that the annual demand is 1000 units. Perform a simulation for different order sizes (from 10 per order to 1000 units per order).

Solution using base R:

```
==========
set.seed(123)

order_cost <- 50
holding_cost <- 2
annual_demand <- 1000

total_cost_for_order_size <- function(order_size)  {
  total_cost <- (annual_demand / order_size) * order_cost
  + (order_size / 2) * holding_cost
  return(total_cost)
}

order_sizes <- seq(10, 1000, by = 10)
total_costs <- numeric(length(order_sizes))

for (i in seq_along(order_sizes))  {
  total_costs[i] <- total_cost_for_order_size(order_sizes[i])
}

order_sizes[which.min(total_costs)]

==========
Expected output (optimum order size to minimize total costs)
[1] 220
==========
```

Interpretation

The optimal order size for reagents is 220 units per order, minimizing total costs while ensuring sufficient supply.

Explanation and notes

This R code determines the optimal order size that minimizes the total cost of managing inventory. The optimal order size is calculated by evaluating the trade-off between ordering costs and holding costs for different order sizes.

The code begins by setting a random seed using set.seed(123) for reproducibility.

The fixed parameters include the order cost (order_cost = 50) in thousands of pesos per order, which represents the cost of placing a single order; the holding cost (holding_cost = 2) in thousands of pesos, which represents the cost of holding one unit of inventory annually; and the annual demand (annual_demand = 1000) in units of PCR cartridges, the total number of units required per year.

The function total_cost_for_order_size calculates the total inventory management cost for a given order size. It computes the total cost as the sum of:

1. The ordering cost: (annual_demand / order_size) * order_cost, which depends on how many orders need to be placed annually.
2. The holding cost: (order_size / 2) * holding_cost, which represents the cost of holding the average inventory level (assumed to be half the order size).

Next, a sequence of potential order sizes is generated using seq(10, 1000, by = 10), representing possible quantities from 10 to 1000 units in increments of 10.

An empty vector, total_costs, is initialized to store the total cost for each order size. A for loop iterates through all possible order sizes, calculates the total cost for each using the optimal_order_quantity function, and stores the result in total_costs.

Finally, the code identifies the order size with the minimum total cost using order_sizes[which.min(total_costs)].

This value corresponds to the EOQ, which is the order size that minimizes the combined ordering and holding costs. The result provides a practical decision point for optimizing inventory management.

DISCUSSION

Benefits of using base R for simulations

Accessibility

Base R is inherently accessible, as it is included in every R installation. Users do not need to install or learn additional packages, making it ideal for those new to R or working in environments with limited internet access or system permissions. This simplicity reduces the setup time and ensures that simulations can be executed without technical barriers, streamlining the learning and implementation process for operations research (OR) applications.

Flexibility

Base R's built-in functions and loop structures provide immense flexibility, allowing users to create tailored solutions for specific OR problems. Unlike pre-built packages, which often require users to adapt their problems to fit the package's framework, base R enables the customization of code to address unique scenarios. For example, using for loops and conditional statements, users can model workflows, simulate inventory dynamics, or perform Monte Carlo experiments with parameters that closely match their operational realities.

Transparency

One of the major strengths of using base R is the explicit and transparent nature of the code. Each step of the simulation process is visible and traceable, making it easier to understand the logic behind the calculations. This transparency is particularly valuable in educational settings, where understanding the underlying processes is just as important as obtaining results. For instance, educators can use base R to demonstrate the mechanics of queuing theory or inventory modeling, breaking down complex concepts into manageable steps.

Performance for simple tasks

Base R performs well for small-scale problems, where the computational demands are moderate. For tasks like simulating sample arrival times, calculating resource utilization, or testing simple queuing models, base R executes efficiently without the overhead of loading and running additional libraries. This makes it a practical choice for quick, straightforward simulations that do not require high levels of complexity or scalability.

Limitations and contextual suitability

Scalability challenges

While base R is sufficient for simple OR simulations, its performance can decline when handling large datasets or complex models. Loops in base R, such as for or while, are not optimized for high scalability and may result in slower execution compared to vectorized operations or specialized simulation libraries. For example, simulating thousands of events in a discrete-event simulation might require significant computational time, making package-based solutions like simmer or parallelized approaches more suitable.

Readability concerns

As simulations grow in complexity, the verbosity of base R code can make scripts harder to read and maintain. Unlike specialized libraries that encapsulate intricate operations into single functions, base R requires users to write out each step explicitly. This verbosity can lead to longer scripts that are challenging to debug and interpret, especially for collaborative projects or long-term use.

Lack of specialized features

Base R lacks the advanced functionalities provided by dedicated simulation packages. Features such as parallel processing, pre-built queuing models, and tools for stochastic optimization are not available natively. For instance, simulating real-time laboratory workflows or performing agent-based modeling may require external packages like simmer or SimPy for efficiency and scalability. This limitation makes base R less suitable for highly dynamic or large-scale OR problems.

Balancing benefits and limitations

Simplicity vs. complexity

For basic OR simulations, the simplicity and accessibility of base R often outweigh its limitations. Tasks such as modeling a single-server queuing system or simulating demand fluctuations can be efficiently accomplished using base R's straightforward constructs. However, as the complexity of the problem increases, the advantages of using specialized packages become more apparent. For instance, a laboratory workflow involving multiple servers, priority queues, and stochastic elements would be better addressed with dedicated simulation tools.

Context-driven suitability

The suitability of base R for OR simulations depends heavily on the context. In resource-limited environments or for educational purposes, the transparency and accessibility of base R make it an excellent choice. Conversely, in high-volume industrial settings or research scenarios requiring advanced modeling, the lack of scalability and specialized features may hinder its applicability. In such cases, integrating base R with additional libraries or external software might provide a balanced solution.

Success and utility of simulation depend on accuracy of assumptions

The success and utility of computer-assisted simulations in operations research processes are heavily dependent on accurately modeling the stochastic nature of the phenomena under consideration. Accurate modeling of stochastic processes is essential, as errors in these models can lead to significant consequences, including economic losses and safety risks.[65] Techniques such as observation-enhanced verification and probabilistic model checking have been developed to improve the accuracy of models by incorporating real-world data, thereby refining continuous-time and discrete-time Markov models to better capture process behaviors.[65]

CONCLUSION

In this paper we demonstrated a simple step by step workflow to analyze operations research type of questions arising from clinical laboratory management scenarios.

We showed that even just Base R provides a powerful, accessible, and flexible platform for performing simple operations research (OR) simulations, offering a straightforward framework for modeling diverse scenarios such as queue simulations, inventory management, and basic workflow optimization. Its transparency and ease of use make it particularly well-suited for small-scale problems, educational purposes, or situations where simplicity and minimal setup are priorities. For beginners or users in resource-constrained environments, base R serves as an effective tool to explore OR concepts and conduct meaningful analyses without relying on additional libraries.

However, its limitations in scalability, readability, and advanced features necessitate careful evaluation of the task's complexity. While base R excels in leveraging R's core statistical and mathematical modeling strengths, users must recognize when transitioning to specialized tools is required for addressing more complex OR challenges. Understanding its unique strengths and constraints allows users to harness base R effectively while building a solid foundation for advancing to more robust methodologies.

STATEMENT OF AUTHORSHIP

The authors certified fulfillment of ICMJE authorship criteria.

AUTHOR DISCLOSURE

The authors declare no conflict of interest and associated funding.

FUNDING SOURCE

None.

REFERENCES

1. Banks J, Carson JS, Nelson BL, David MN. Discrete-event system simulation. 5th ed. Upper Saddle River: Prentice Hall; 2010.
2. Green LV. Using operations research to reduce delays for healthcare. In: Chen ZL, Raghavan S, Gray P, Greenberg HJ, eds. State-of-the-art decision-making tools in the information-intensive age. INFORMS. DOI: 10.1287/educ.1080.0049

3. Green LV. OM Forum—The vital role of operations analysis in improving healthcare delivery. M&SOM, INFORMS. 2012;14(4):488–94. DOI: 10.1287/msom.1120.0397
4. Ucar I, Smeets B, Azcorra A. simmer: discrete-event simulation for R. J Stat Softw. 2019;90(2):1-30. DOI: 10.18637/jss.v090.i02
5. Dibba F, Herulambang W, Setyatama F. Reagent stock prediction using Monte Carlo method at Populer Clinical Laboratory Surabaya. J Electr Eng Comput Sci. 2023;7(2):1329–36. DOI: 10.54732/jeecs.v7i2.27
6. Hussain MR, Hussain ME. A new neuroinformatics approach to optimize diagnosis cost in neurology: an operational research tool. iJOE. 2019;15(06):31-52. DOI: 10.3991/ijoe.v15i06.10141
7. Khan M, Khalid P, Almorsy L, Khalifa M. Improving timeliness of diagnostic healthcare services: effective strategies and recommendations. Stud Health Technol Inform. 2016;226:183-6. PMID: 27350499
8. Pierskalla WP, Brailer DJ. Chapter 13 Applications of operations research in health care delivery. In: Handbooks in operations research and management science. 1994;6:469-505. DOI: 10.1016/S0927-0507(05)80094-5
9. Bodtker K, Wilson L, Godolphin W. Simulation modelling to assist operational management and planning in clinical laboratories. Simulation. 1993;60(4):247–55. DOI:10.1177/003754979306000405
10. Vogt W, Braun SL, Hanssmann F, et al. Realistic modeling of clinical laboratory operation by computer simulation. Clin Chem. 1994;40(6):922–8. PMID: 8087987
11. Dawande PP, Wankhade RS, Akhtar FI, Noman O. Turnaround Time: an efficacy measure for medical laboratories. 2022;14(9):e28824. PMID: 36225468 PMCID: PMC9535613 DOI: 10.7759/cureus.28824
12. Bowles J, Czekster RM, Redeker G, Webber T. A simulation study on demand disruptions and limited resources for healthcare provision. In: Lecture notes in computer science; 2021.
13. Mannello F, Plebani M. Current issues, challenges, and future perspectives in clinical laboratory medicine. J Clin Med. 2022 ;11(3):634. PMID: 35160086 PMCID: PMC8836853 DOI: 10.3390/jcm11030634
14. Ritika Goel, Tanya Karn, Rahul Kushwaha, Ashima Mehta. Healthcare resource allocation optimization. IJARSCT. 2024;4(5):429–33. https://ijarsct.co.in/Paper17569.pdf
15. Lippi G, Simundic AM, Plebani M. Phlebotomy, stat testing and laboratory organization: an intriguing relationship. Clin Chem Lab Med. 2012;50(12):2065–8. PMID: 23093204 DOI: 10.1515/cclm-2012-0374
16. Wang J, Wang J. Real-Time adaptive allocation of emergency department resources and performance simulation based on stochastic timed petri nets. IEEE Trans Comput Soc Syst. 2023;10(4):1986–96. DOI: 10.1109/TCSS.2023.3266501.
17. Wicaksono PA, Sutrisno S, Solikhin S, Aziz A. Optimizing production planning and inventory management in post-pandemic recovery using a multi-period hybrid uncertain optimization model. RAIRO Oper Res. 2024;58(5):3805–21. DOI: 10.1051/ro/2024136
18. Boche B, Temam S, Kebede O. Inventory management performance for laboratory commodities and their challenges in public health facilities of Gambella Regional State, Ethiopia: a mixed cross-sectional study. Heliyon. 2022;8(11):e11357. PMID: 36387489 PMCID: PMC9649967 DOI: 10.1016/j.heliyon.2022.e11357
19. Kafoe AS. Supply chain resilience strategy for healthcare organizations: crucial steps in addressing the impact of natural Disasters. In: Burrell DN, editor. Advances in Logistics, Operations, and Management Science. IGI Global; 2024. DOI: 10.4018/979-8-3693-4288-6.ch001
20. Ofaka, Enyanwu C, Daniel NW, Sonika, Goel R. Effect of pre-analytical errors in laboratory testing facilities: the way forward. TIJPH. 2023;11(2):15–21. DOI: 10.21522/TIJPH.2013.11.02.Art002
21. Sharma D, Cotton M. Overcoming the barriers between resource constraints and healthcare quality. Trop Doct. 2023;53(3):341–3. PMID: 37366617 DOI: 10.1177/00494755231183784
22. Lippi G, Cadamuro J, Danese E, et al. Disruption of laboratory activities during the COVID-19 Pandemic: results of an EFLM Task Force Preparation of Labs for Emergencies (TF-PLE) survey. 2023;34(3):213-9. PMID: 37868082 PMCID: PMC10588075
23. Huq Ronny FM, Sherpa T, Choesang T, Ahmad S. Looking into the Laboratory Staffing Issues that Affected Ambulatory Care Clinical Laboratory Operations during the COVID-19 Pandemic. Lab Med. 2023;54(4):e114–6. PMID: 36282479 PMCID: PMC9620378 DOI: 10.1093/labmed/lmac139
24. Liu K, Liu C, Xiang X, Tian Z. Testing facility location and dynamic capacity planning for pandemics with demand uncertainty. Eur J Oper Res. 2023;304(1):150–68. PMID: 34848916 PMCID: PMC8613006 DOI: 10.1016/j.ejor.2021.11.028
25. Bean L, Rosendorff A, Dallaire S, et al. eP321: Extending and adapting the functions of genetic laboratories in the continuing COVID pandemic–challenges and successes. Genet Med. 2022;24(3):S200–1. PMCID: PMC8935068 DOI: 10.1016/j.gim.2022.01.356
26. Wilson ML. Decreasing inappropriate laboratory test utilization: controlling costs and improving quality of care. Am J Clin Pathol. 2015;143(5):614–6. DOI: 10.1309/AJCPHQODM9XYWLZ9
27. Bello S, Adebowale AS, Dairo MD, et al. Strategies for rapid scale up of laboratory capacity in a public health emergency in a resource-constrained setting: the SARS-CoV-2 Nigeria response experience. 2023. DOI: 10.21203/rs.3.rs-2897840/v1 [preprint]
28. Paul II, Victoria M, Olalere OS. Patient turnaround time: concern of medical laboratory scientist. SJMLS. 2023;8(1):96–107. DOI: 10.4314/sokjmls.v8i1.12
29. Sibley J, Quellie SG, Mendy PO, et al. Integration of quality management systems in a rural, low-resource environment: the experience at Phebe Hospital in Bong County, Liberia. GJOBOH. 2022;1(2):1-7. DOI: 10.36108/GJOBOH/2202.10.0210
30. Arbiol-Roca A, Dot-Bach D. Critical issues and new trends on stat tests in clinical laboratory. EJIFCC. 2019;30(1):59–66. PMID: 30881275 PMCID: PMC6416811

31. Amilal Kulhari. Significance of linear programming for optimization. IJARSCT. 2023;3(15):179–86.
32. Wang L, Zhao J. Mathematical optimization. In: Wang L, Zhao J, editors. Architecture of advanced numerical analysis systems: designing a scientific computing system using OCaml. Berkeley, CA: Apress; 2023. DOI: 10.1007/978-1-4842-8853-5_4.
33. Chanda R, Pabalkar V, Gupta S. A study on application of linear programming on product mix for profit maximization and cost optimization. Indian J Sci Technol. 2022;15(22):1067–74. DOI: 10.17485/IJST/v15i22.164
34. Vázquez-Serrano JI, Peimbert-García RE, Cárdenas-Barrón LE. Discrete-event simulation modeling in healthcare: a comprehensive review. Int J Environ Res Public Health. 2021;18(22):12262. PMID: 34832016 PMCID: PMC8625660 DOI: 10.3390/ijerph182212262
35. Zhang X. Application of discrete event simulation in health care: a systematic review. BMC Health Serv Res. 2018;18(1):687. PMID: 30180848 PMCID: PMC6123911 DOI: 10.1186/s12913-018-3456-4
36. Moyi AU, Gamagiwa KB, Ibidoja OJ, Muhammad G. Application of Queuing Theory in a University Clinic. Int J Sci Glob Sustain. 2022;8(1):1–9. DOI: 10.57233/ijsgs.v8i1.338
37. Yadav SK, Singh G, Sarin N, Singh S, Gupta R. Optimization of manpower deployment for COVID-19 screening in a tertiary care hospital: a study of utility of queuing analysis. Disaster Med Public Health Prep. 2022;16(6):2388–92. PMID: 34284837 PMCID: PMC8438508 DOI: 10.1017/dmp.2021.228
38. Metropolis N, Ulam S. The Monte Carlo method. J Am Stat Assoc. 1949;44(247):335–41.
39. Robert C, Casella G. A short history of Markov Chain Monte Carlo: subjective recollections from incomplete data. Statist Sci. 2011;26(1):102-15. DOI: 10.1214/10-STS351
40. Badika EM, Zyryanov DA, Babchinetsky SG. Using the python programming language in simulation modeling. CJ. 2022;7(6(68)):18–23.
41. Staples TL. Expansion and evolution of the R programming language. R Soc Open Sci. 2023;10(4):221550. PMID: 37063989 PMCID: PMC10090872 DOI: 10.1098/rsos.221550
42. Hoffmann GE. Simulation-based reorganization and automation planning in clinical laboratories. J Lab Autom. 1998;3(2):31–3. DOI: 10.1177/221106829800300209
43. Westgard JO. Simulation and modeling for optimizing quality control and improving analytical quality assurance. Clin Chem. 1992;38(2):175–8. PMID: 1540997
44. Raimbault J, Pumain D. Methods for exploring simulation models. In: Geographical modeling: cities and territories, vol. 2. John Wiley & Sons, Ltd; 2019. DOI: 10.1002/9781119687290.ch5.
45. Rece L, Vlase S, Ciuiu D, Neculoiu G, Mocanu S, Modrea A. Queueing theory-based mathematical models applied to enterprise organization and industrial production optimization. Mathematics. 2022;10(14):2520.
46. Sotomayor NAO, Fiestas LAZ, Vergaray EF. Optimization software in operational research analysis in a public university. Rev Geintec. 2021;11(4):3061–79. https://revistageintec.net/old/wp-content/uploads/2022/03/2350.pdf
47. Castillo I, Lee T, Pinter J. Integrated software tools for the OR/MS Classroom. Algorithmic Oper Res. 2008;3(1). https://journals.lib.unb.ca/index.php/AOR/article/view/1250.
48. Kaseko MS. Comparative evaluation of simulation software for traffic operations; 2002. https://www.nevadadot.com/home/showdocument?id=4010
49. Hlupic V. Simulation software: an Operational Research Society survey of academic and industrial users. In: 2000 Winter Simulation Conference Proceedings (Cat No00CH37165). Orlando, FL, USA: IEEE; 2000. DOI: 10.1109/WSC.2000.899156
50. Kruger PS. Micro-computer simulation software: a review. ORiON. 1986;2(1):1-14. DOI: 10.5784/2-1-504
51. Dunning I, Gupta V, King A, Kung J, Lubin M, Silberholz J. A course on advanced software tools for operations research and analytics. ITE. 2015;15(2):169–79. DOI: 10.1287/ited.2014.0131
52. Archetti C, Speranza MG, Garrafa E. Managing an automated clinical laboratory: optimization challenges and opportunities. EJDP. 2020;8(1):41–60. DOI: 10.1007/s40070-019-00097-2
53. Gungoren MS. Crossing the chasm: strategies for digital transformation in clinical laboratories. Clin Chem Lab Med. 2023;61(4):570–5. PMID: 36753305 DOI: 10.1515/cclm-2022-1229
54. Wheeler SE, Block DR, Bunch DR, et al. Clinical laboratory informatics and analytics: challenges and opportunities. Clin Chem. 2022;68(11):1361–7. PMID: 36264683 DOI: 10.1093/clinchem/hvac157
55. Ong SK, Donovan GT, Ndefru N, et al. Strengthening the clinical laboratory workforce in Cambodia: a case study of a mixed-method in-service training program to improve laboratory quality management system oversight. Hum Resour Health. 2020;18(1):84. PMID: 33148269 PMCID: PMC7610006 DOI: 10.1186/s12960-020-00521-8
56. Crawley MJ. The R Book, 1st ed. Wiley; 2007. DOI: 10.1002/9780470515075
57. Anderson TR. Optimization modeling using R, 1st ed. Boca Raton: Chapman and Hall/CRC; 2022. DOI: 10.1201/9781003051251
58. Erdem Kara B. Computer adaptive testing simulations in R. Int J Assess Tool Educ. 2019;6(5):44–56. DOI: 10.21449/ijate.621157
59. Hair JF, Hult GTM, Ringle CM, Sarstedt M, Danks NP, Ray S. Overview of R and RStudio. In: Partial least squares structural equation modeling (PLS-SEM) using R. Cham: Springer International Publishing; 2021. https://link.springer.com/10.1007/978-3-030-80519-7_2.
60. Zhang X, Maas Z. Using R as a simulation tool in teaching introductory statistics. Int Elect J Math Ed. 2019;14(3):599-610. DOI: 10.29333/iejme/5773
61. Hofert M, Mächler M. Parallel and other simulations in R made easy: an end-to-end study. J Stat Soft. 2016;69(4):1-44. DOI: 10.18637/jss.v069.i04
62. Wang W, Hallow K, James D. A Tutorial on RxODE: simulating differential equation pharmacometric models in R. CPT Pharmacometrics Syst Pharmacol. 2016;5(1):3–10. PMID: 26844010 PMCID: PMC4728294 DOI: 10.1002/psp4.12052

63. R Core Team. R: a language and environment for statistical computing (v. 4.4.2). R Foundation for Statistical Computing, Vienna, Austria; 2024. https://cran.r-project.org/doc/manuals/r-release/fullrefman.pdf
64. Posit team. RStudio: integrated development environment for R. Posit Software, PBC, Boston, Massachusetts; 2024. http://www.posit.co/
65. Paterson C. Observation-enhanced verification of operational processes. PhD thesis, University of York; 2018. https://etheses.whiterose.ac.uk/23257/