Final Report of Traineeship Program 2023

On

"Classify Suspected Infection"

MEDTOUREASY

28th July 2023

Submitted By:

Devashish Chadha

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The traineeship opportunity that I, Devashish Chadha had with MedTourEasy was a great chance for learning and understanding the intricacies of the subject of Data visualization in Data Analytics; and also, for personal as well as professional development. I am very obliged for having a chance to interact with so many professionals who guided me throughout the traineeship project and made it a great learning curve for me.

Firstly, I express my deepest gratitude and special thanks to the Training & Development Team of MedTourEasy who allowed me to carry out my traineeship at their esteemed organization. Also, I express my thanks to the team for making me understand the details of the Data Analytics profile and training me in the same so that I can carry out the project properly and with maximum client satisfaction .

I would also like to thank my Mentor and my colleagues who made the working environment productive and very conducive.

ABSTRACT

Sepsis is a serious condition that happens when the body's immune system has an extreme response to an infection. The body's reaction causes damage to its tissues and organs. Sepsis can affect anyone, but people who are older, very young, pregnant, or have other health problems are at higher risk. Common signs of sepsis include fever, fast heart rate, rapid breathing, confusion, and body pain. It can lead to septic shock, multiple organ failure, and death. Sepsis is usually caused by bacterial infections but may be the result of other infections such as viruses, parasites, or fungi. Treatment for sepsis requires medical care. It will include antimicrobials, intravenous fluids, and careful monitoring.

Implementing preventive measures against infections, such as good hygiene practices, ensuring access to vaccination programs, improved sanitation, water quality, and availability, and other infection prevention and control best practices both in the community and health care settings, are key steps in reducing the occurrence of sepsis. Early diagnosis and timely and appropriate clinical management of sepsis, such as optimal antimicrobial use and fluid resuscitation, is crucial to increase the likelihood of survival. Even though the onset of sepsis can be acute and poses a short-term mortality burden, it can also be the cause of significant long-term morbidity requiring treatment and support. Thus, sepsis requires a multidisciplinary approach.

My project focuses on identifying hospital patients with severe infections using medical record data. To make this project successful the language used is "R" which uses the data. table package that further includes the use of various functions like shift, merge, fread, and many more.

I. INTRODUCTION

1.1 About the Company:

MedTourEasy, a global healthcare company, provides you with the informational resources needed to evaluate your global options. It helps you find the right healthcare solution based on specific health needs, and affordable care while meeting the quality standards that you expect to have in healthcare. MedTourEasy improves access to healthcare for people everywhere. It is an easy-to-use platform and service that helps patients to get medical second opinions and to schedule affordable, high-quality medical treatment abroad.

1.2 About the Project:

Sepsis (septicaemia in British English), or blood poisoning, is a life-threatening condition that arises when the body's response to infection causes injury to its tissues and organs.

Disease severity partly determines the outcome. The risk of death from sepsis is as high as 30%, while for severe sepsis it is as high as 50% and for septic shock 80%. Sepsis affected about 49 million people in 2017, with 11 million deaths (1 in 5 deaths worldwide). In the developed world, approximately 0.2 to 3 people per 1000 are affected by sepsis yearly, resulting in about a million cases per year in the United States. Rates of disease have been increasing. Some data indicate that sepsis is more common among males than females, however, other data show a greater prevalence of the disease among women. Descriptions of sepsis date back to the time of Hippocrates.

In that reference, it is extremely crucial to identify the patients suffering from SEPSIS using medical records to analyze the data and takes steps to fight against this dreadful disease. Additionally, MedTourEasy, being one of the globally upcoming telemedicine companies in global healthcare, the firm needs to understand the current situation of SEPSIS to gain more insights into the intensity of this disease, the symptoms of the patients, and the impact it will have worldwide. Also, depending on the results of the analysis, this may be used for increasing their market presence and capacity planning.

Hence, the project aims to analyze medical record data to gain insights into the prevalence of severe infections among hospital patients and to identify potential cases of sepsis. By leveraging the data. table package in R, the analysis will involve exploring antibiotic and blood culture data, understanding treatment patterns, and examining the timing of events related to antibiotic administration and blood culture collection

1.3 Objectives and Deliverables:

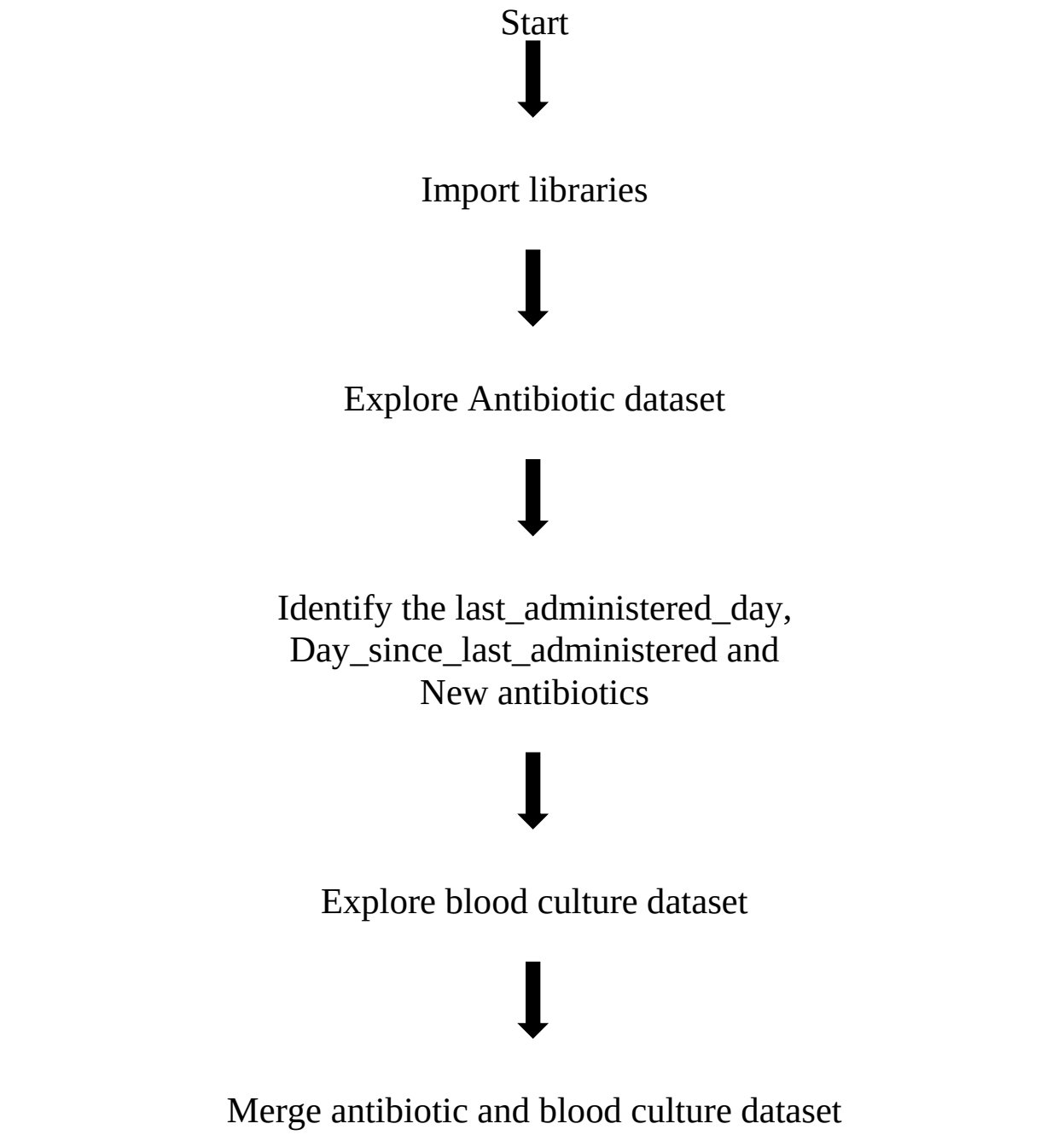
The main objective of this project is to identify hospital patients with severe infection, particularly sepsis, through an analysis of medical record data. The specific tasks involved in achieving this objective include:

- Exploring and understanding the antibiotic data related to patient treatment.
- Identifying "new" antibiotics used in patient treatment based on certain criteria.
- Investigating the blood culture data to gain insights into patient infections.
- Merging the antibiotic and blood culture data to establish relevant associations.
- Determining whether antibiotic administration and blood culture timing align within a specific window.
- Identifying patients who received intravenous (I.V.) antibiotics within the +/-2 day window of blood culture.
- Locating the first day of potential 4-day antibiotic sequences for each blood culture event.
- Extracting the first four antibiotic days for each patient/blood culture combination.
- Identifying four-day sequences with no skips of more than one day.
- Creating a subset dataset for further analysis based on specific criteria.
- Calculating the percentage of patients with presumed serious infections using the merged data.

II. METHODOLOGY

2.1 Flow of the Project:

The project followed the following steps to accomplish the desired objectives and deliverables. Each step has been explained in detail in the following section.



Link antibiotic and blood culture timing using
Variable drug_in_bcx_window

Identify I.V. Antibiotics in Blood Culture Window

Locate the First Day of Potential 4-Day Antibiotic Sequences

Extract First Four Antibiotic Days

Identify Four-Day Sequences with No Skips

Create a Subset Dataset for Further Analysis

Calculate the Percentage of Patients with Presumed Serious Infections

Stop

2.2 Language and Platform Used:

2.2.1 Language: R

R is a language and environment designed for statistical computing and graphics. It is a GNU project similar to the S language and environment, originally developed at Bell Laboratories by John Chambers and colleagues. R can be seen as an alternative implementation of S. Although there are some important differences, much of the code written for S remains unchanged when used in R.

R offers a diverse range of statistical and graphical techniques, including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, and more. It is a highly extensible platform. The S language is commonly used for statistical methodology research, and R provides an Open-Source path for participating in such activities.

One of R's strengths is its ability to produce well-designed, publication-quality plots with ease, even incorporating mathematical symbols and formulas when necessary. The default graphics settings have been meticulously crafted, while still allowing the user to maintain full control.

R is freely available under the terms of the Free Software Foundation's GNU General Public License in source code form. It can be compiled and run on various UNIX platforms and similar systems (including FreeBSD and Linux), Windows, and MacOS.

R is a comprehensive suite of software tools designed for data manipulation, calculation, and graphical display. It encompasses the following features:

1. An efficient data handling and storage system.
2. A collection of operators for array calculations, with a focus on matrices.
3. A well-organized and integrated set of intermediate tools for data analysis.
4. Graphical capabilities for data analysis and visualization, available both on-screen and in printed formats.
5. A powerful and straightforward programming language, which includes conditionals, loops, user-defined recursive functions, and input/output functionalities.

2.2.2 IDE: RStudio

R Studio is an integrated development environment (IDE) designed for R. An IDE provides a graphical user interface (GUI) that allows users to write code, view results, and inspect variables generated during the programming process.

1. R Studio is offered in both Open Source and Commercial versions, catering to different user needs.
2. Additionally, R Studio is available in both Desktop and Server versions, enabling flexibility in usage.
3. Furthermore, R Studio supports various platforms, including Windows, Linux, and macOS, ensuring broad accessibility.

RStudio is a free and open-source tool that serves as an Integrated Development Environment (IDE) for the R language. Additionally, it offers enterprise-ready professional software for data science teams to collaborate and share their work effectively.

1. The left panel, known as the **console panel**, is where users interact with R by inputting commands and viewing the generated results.

2. To the top right, the Environmental/History panel is located, featuring two tabs:

    a. **Environment tab**: This tab displays the temporary workspace and the variables created during the programming process.
    b. **History tab**: Here, users can view a history of all the commands used since the start of R Studio usage.

3. In the right bottom panel, several tabs are available, including files, plots, packages, help, and viewer. Each tab serves specific functions, enhancing the user experience and ease of data analysis.

2.2.3 Package: data.table

Data. table is a powerful extension of the data. frame package in R. Its primary application is for efficiently handling large datasets, providing speedy aggregation, rapid addition, update, and removal of columns, and quick ordered joins. Additionally, data. table boasts a fast file reader, making it highly efficient for working with substantial data.

One of the key advantages of data. table is its flexible and intuitive syntax, which accelerates the development process. Moreover, data. table offers notable features such as fast primary-ordered indexing and automatic secondary indexing. These capabilities, combined with a memory-efficient combined join and group by operations, contribute to its performance and popularity for data manipulation tasks.

Installation:

```
install.packages("data.table")
```

Loading:

```
library(data.table)
```

2.2.4 Functions used:

1. **shift():**

   The data. table package in R provides a convenient method for sub-setting data and creating new columns using the shift() function. The shift() function allows elements within a column to be moved either in the forward direction (lead) or the backward direction (lag) based on a specified number of steps.

   The general syntax for using the shift() function is as follows:

   shift(x, n = 1L, type = c("lag", "lead"), fill = NA, by = NULL)

Below is the implementation of the shift() function in this project:

```
antibioticDT[,last_administration_day := shift(day_given,n=1,fill = NA,type= "lag")
             ,by = .(patient_id,antibiotic_type)]
```

2. fread():

The data. table package in R is well-known for its efficiency and user-friendly features, particularly for managing extensive datasets. Among its valuable functions is fread(), which stands out as a specialized tool for efficiently importing data from regular delimited files directly into R without any unnecessary complexities.

Key Features and Advantages of fread():

1. Speed and Efficiency: fread() excels in reading and parsing data swiftly. Leveraging a highly optimized C++ implementation, it becomes the preferred choice for dealing with large datasets, ensuring speedy data processing.

2. Automatic Data Type Inference: The fread() function exhibits intelligent data type inference. It automatically determines the data types for each column in the imported file, eliminating the need for explicit data type specifications by users.

3. Support for Delimited Files: fread() seamlessly handles various delimiter-separated files, including CSV (Comma-Separated Values), TSV (Tab-Separated Values), and other customized delimiters, making it highly versatile.

4. Effortless Handling of Large Datasets: With its remarkable efficiency in managing substantial datasets, fread() becomes the preferred choice for data professionals who routinely work with vast amounts of information.

General syntax of fread():

fread(input, data.table = FALSE, sep = ",", header = "auto", sep2 = NULL,
col.names = NULL, colClasses = NULL, nrows = -1, skip = 0, select = NULL,
drop = NULL, verbose = FALSE, autostart = 5, na.strings = "NA",
encoding = "unknown", strip.white = TRUE, blank.lines.skip = TRUE,
skipEmptyLines = TRUE, integer64 = "integer64")

Below is the implementation of fread () in the project:

```
antibioticDT <- fread("C:\\Users\\Devashish Chadha\\OneDrive\\Desktop\\
                      Data anlytics by Google\\Project\\project.data\\
                      Classify Suspected Infection in Patients\\datasets\\
                      antibioticDT.csv")
```

3. by = .():

In R, the by() function is used for grouping data and performing operations within each group. It is commonly used in combination with functions like aggregate(), summarize(), transform(), or when working with the data. table package**.**

General syntax of by = .() function:

by(data, INDICES, FUN, ...)

Below is the implementation of by = .() in the project:

```
merged_df <- merge(antibioticDT,blood_cultureDT,by = "patient_id",all = FALSE)
```

4. **: = (assign operator):**
   The assign operator is used to create or modify columns within a data. table. It is used for in-place updates to the data.

   The general syntax of: = (assign operator) :

   data.table[, new_column := expression]

   Below is the implementation of: = (assign operator) in the project:

```
new_dt[,num_antibiotic_days := .N, by = .(patient_id,blood_culture_day)]
```

5. **merge():**
   While data. table has its efficient join syntax, you can also use the merge() function for merging data. tables based on common columns. The R merge function allows merging two data frames by common columns or by row names. This function allows you to perform different database (SQL) joins, like left join, inner join, right join, or full join, among others.

   General syntax of merge() :

   merge(x, y, by = NULL, by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all, sort = TRUE, suffixes = c(".x", ".y"), ...)

   Below is the implementation of this function in this project:

```
merged_df <- merge(antibioticDT,blood_cultureDT,by = "patient_id",all = FALSE)
```

6. **unique():**

   The unique() function is used to extract unique rows from data. table based on specified columns.

   The unique() function in R serves the purpose of identifying and eliminating duplicate values or rows from vectors, data frames, or matrices. It plays a crucial role in Exploratory Data Analysis (EDA) as it allows for the direct detection and removal of duplicates in the dataset.

   General syntax of unique():

   Df <- unique(Df)

   Below is the implementation of this function in this project:

```
suspected_infection <- unique(suspected_infection)
```

III. IMPLEMENTATION

3.1 Gathering Requirements and Defining Problem Statement:

This is the first step wherein the requirements are collected from the clients to understand the deliverables and goals to be achieved after which a problem statement is defined which has to be adhered to while developing the project.

3.2 Data Collection and Importing:

Data collection is a systematic approach for gathering and measuring information from a variety of sources to obtain a complete and accurate picture of an interesting area. It helps an individual or organization to address specific questions, determine outcomes and forecast future probabilities and patterns. The data collected and imported is provided by the MedTourEasy organization. The data imported contains three datasets “all_patients.csv”, “antibioticDT.csv” and “blood_cultureDT.csv”.

Data importing is referred to as uploading the required data into the coding environment from internal sources (computer) or external sources (online websites and data repositories). This data can then be manipulated, aggregated, and filtered according to the requirements and needs of the project.

Packages Used:

Fread(): The data. table package in R is well-known for its efficiency and user-friendly features, particularly for managing extensive datasets. Among its valuable functions is fread(), which stands out as a specialized tool for efficiently importing data from regular delimited files directly into R without any unnecessary complexities.

```
Sample Code:


install.packages("data.table")

library(data.table)

antibioticDT <- fread("C:\\Users\\Devashish Chadha\\OneDrive\\Desktop\\Data
anlytics by Google\\Project\\project.data\\Classify Suspected Infection in
Patients\\datasets\\antibioticDT.csv")
```

3.3 Designing Databases:

Once the data has been collected and imported into the R environment, it is important to design the structure of the database tables to identify the constraints in the data, keys, dependencies, and relations between various tables.

Once the data is imported into the environment, it is converted into a data frame (datatype in R) which makes it easy to maintain the data in the form of tables. The various tables which have been created are mentioned as follows:

1. antibioticDT:

Attributes	**Datatype**
patient_id	Int
day_given	Int
antibiotic_type	Chr
route	Chr
last_administration_day	Int
days_since_last_admin	Int
antibiotic_new	Num

2. blood_cultureDT:

Attributes	**Datatype**
patient_id	Int
blood_culture_day	Int

3. all_patients.DT:

Attributes	**Datatype**
patient_id	Int
infection	Num

3.4 Data Clean :

Clean data refers to data that has been thoroughly reviewed, processed, and corrected to ensure its accuracy, consistency, and reliability. Clean data is free from errors, inconsistencies, and missing values that could potentially impact the quality and validity of data analysis and decision-making processes.

Data cleaning, also known as data cleansing or data scrubbing, is the process of identifying and correcting errors, inconsistencies, and inaccuracies in a dataset to improve its quality and reliability. It involves various tasks, such as handling missing values, removing duplicates, transforming data types, dealing with outliers, and resolving inconsistencies.

Concerning the various datasets used, there may be many NA or duplicate values in our datasets which are to be removed to get unbiased results. The various techniques used in this project are as follows:

Packages Used:

Data.table():

Data. table is a powerful extension of the data. frame package in R. Its primary application is for efficiently handling large datasets, providing speedy aggregation, rapid addition, update, and removal of columns, and quick ordered joins. Additionally, data. table boasts a fast file reader, making it highly efficient for working with substantial data.

Functions Used:

1. Is.na():

In R, the is.na() function is used to identify missing or NA (Not Available) values in a vector, matrix, data frame, or any other data structure. The function returns a logical vector of the same length as the input, where each element is TRUE if the corresponding element in the input is NA and FALSE otherwise.

Below is the implementation of it. na in this project:

```
all_patientsDT <- all_patientsDT[is.na(infection), infection := 0]
```

2. **Unique():**
   The unique() function is used to extract unique rows from data. table based on specified columns. The unique() function in R serves the purpose of identifying and eliminating duplicate values or rows from vectors, data frames, or matrices. It plays a crucial role in Exploratory Data Analysis (EDA) as it allows for the direct detection and removal of duplicates in the dataset.

   Below is the implementation of unique() in this project:

```
suspected_infection <- unique(suspected_infection)
```

3. **As.numeric():**
   In R, the as. numeric() function is used to convert objects of various types into numeric (i.e., numeric data type) objects. It is commonly used for data type conversion when dealing with numerical data.

   Below is the implementation of as. numeric() in this project:

```
first_four_rows[,four_in_seq := as.numeric(max(diff(day_given)) < 3), by = .(patient_id,blood_culture_day)]
```

4. **shift():**

The data. table package in R provides a convenient method for sub-setting data and creating new columns using the shift() function. The shift() function allows elements within a column to be moved either in the forward direction (lead) or the backward direction (lag) based on a specified number of steps.

The general syntax for using the shift() function is as follows:

shift(x, n = 1L, type = c("lag", "lead"), fill = NA, by = NULL)

Below is the implementation of the shift() function in this project:

```
antibioticDT[,last_administration_day := shift(day_given,n=1,fill = NA,type= "lag")
             ,by = .(patient_id,antibiotic_type)]
```

3.5 Data Filtering:

Filtered data refers to a subset of a larger dataset that has been extracted or selected based on specific criteria or conditions. When working with large datasets, it is common to filter the data to focus on specific observations or records that are of interest for further analysis or reporting. Data filtering is a crucial step in data analysis, as it allows analysts to zoom in on relevant information and discard unnecessary or irrelevant data.

Data filtering is the process of selecting a subset of data from a larger dataset based on specific criteria or conditions. It involves extracting and retaining only the relevant observations or records that meet certain requirements while excluding or omitting data that does not match the specified conditions. Data filtering is an essential step in data analysis, as it allows analysts to focus on the most relevant and meaningful information within a dataset.

Concerning the project, the following data filtering techniques are used:

1. **.() Function:**

   The . () function is used to evaluate expressions within the data. table environment. It allows you to filter data based on more complex conditions involving multiple columns.

   Implementation:

```
suspected_infection <- suspected_infection[,.(patient_id)]
```

2. **.SD symbol:**

   The.SD symbol represents the Subset of Data. table and is used to perform operations on each group of data defined by the argument. Although.SD itself is not a filtering function, it can be combined with other filtering methods to perform group-wise filtering.

   Implementation:

```
first_four_rows <- new_dt[, .SD[1:4], by = .(patient_id,blood_culture_day)]
```

3. **.N symbol:**

   The. N symbol returns the number of rows in each group when used with the by argument. This can help filter groups based on group size.

   Implementation:

```
new_dt[,num_antibiotic_days := .N, by = .(patient_id,blood_culture_day)]
```

Packages Used:

Data. table:

Data. table is a powerful extension of the data. frame package in R. Its primary application is for efficiently handling large datasets, providing speedy aggregation, rapid addition, update, and removal of columns, and quick ordered joins. Additionally, data. table boasts a fast file reader, making it highly efficient for working with substantial data.

IV. SAMPLE SCREENSHOTS AND OBSERVATIONS

4.1. Use of data.table Package:

The project extensively utilizes the data. table package, which is known for its efficiency and speed in handling large datasets. The data. table package provides functionalities like:= for assignment, by grouping aggregations, and shift for lag/lead operations, which are essential for this project.

4.2. Dataset Exploration:

The project begins with exploring and understanding the structure of the antibiotic data and blood culture data. It is crucial to have a good grasp of the dataset before proceeding with any data manipulation tasks.

On viewing the first few rows of the antibiotics data:

	patient_id	day_given	antibiotic_type	route
1	1	2	ciprofloxacin	IV
2	1	4	ciprofloxacin	IV
3	1	6	ciprofloxacin	IV
4	1	18	ciprofloxacin	IV
5	1	7	doxycycline	IV
6	1	9	doxycycline	IV
7	1	16	doxycycline	IV
8	1	15	penicillin	IV
9	8	1	doxycycline	PO
10	8	3	doxycycline	IV

On exploring the blood_culture data:

	patient_id	blood_culture_day
1	1	3
2	1	13
3	8	2
4	8	13
5	23	3
6	39	10
7	45	4
8	45	9
9	45	11
10	51	3
11	51	6
12	59	2
13	64	1
14	66	9
15	66	10
16	69	2
17	69	6
18	69	7
19	69	11
20	69	16

On viewing the all_patients data:

	patient_id	infection
1	1	1
2	5	0
3	8	0
4	9	0
5	12	0
6	16	0
7	19	0
8	23	1
9	25	0
10	39	0
11	40	0
12	41	0

4.3 Data Cleaning and Manipulation:

The project involves various data cleaning and manipulation steps. This includes sorting the data, calculating time differences, merging datasets, and creating new variables based on specific conditions.

4.3.1 Step 1:

Initially, the antibiotics data was sorted in order of patient_id, antibiotic_type, and day_given.

4.3.2 Step 2:

A new variable is added in the antibiotics dataset i.e. last_administration day which calculates the last administration day of antibiotics for each patient and antibiotic type.

4.3.3 Step 3:

Another variable day_since_last_admin calculates the number of days since the antibiotic was administered to a patient and is added to the antibiotics dataset.

4.3.4 Step 4:

Another variable antibiotics new is added to the dataset which is initialized to 0 but later changes to 1 if day_since_last_admin is one or two days since the last antibiotic was given.

	patient_id	day_given	antibiotic_type	route	last_administration_day	days_since_last_admin	antibiotic_new
1	1	2	ciprofloxacin	IV	*NA*	*NA*	1
2	1	4	ciprofloxacin	IV	2	2	0
3	1	6	ciprofloxacin	IV	4	2	0
4	1	18	ciprofloxacin	IV	6	12	1
5	1	7	doxycycline	IV	*NA*	*NA*	1
6	1	9	doxycycline	IV	7	2	0
7	1	16	doxycycline	IV	9	7	1
8	1	15	penicillin	IV	*NA*	*NA*	1
9	8	1	doxycycline	PO	*NA*	*NA*	1
10	8	3	doxycycline	IV	1	2	0

4.3.5 Step 5:

The antibiotics data and the blood_culture data are merged to get better insights and are sorted based on patient_id, blood_culture_day, day_given, and antibiotic_type.

	patient_id	day_given	antibiotic_type	route	last_administration_day	days_since_last_admin	antibiotic_new	blood_culture_day
1	1	2	ciprofloxacin	IV	NA	NA	1	3
2	1	4	ciprofloxacin	IV	2	2	0	3
3	1	6	ciprofloxacin	IV	4	2	0	3
4	1	7	doxycycline	IV	NA	NA	1	3
5	1	9	doxycycline	IV	7	2	0	3
6	1	15	penicillin	IV	NA	NA	1	3
7	1	16	doxycycline	IV	9	7	1	3
8	1	18	ciprofloxacin	IV	6	12	1	3
9	1	15	penicillin	IV	NA	NA	1	13
10	1	16	doxycycline	IV	9	7	1	13

4.4 Logical Data Transformations:

Several logical transformations are performed, such as converting logical values (TRUE/FALSE) to binary (0/1) using as. numeric() function. These transformations are useful in creating indicator variables for further analysis.

The merged dataset is added with a new variable i.e. drug_in_bcx_window that checks if the difference between day_given and blood_culture_day is within 2 days. This variable is set as numeric so as get its results in 1 or 0 i.e. true or false.

	patient_id	day_given	antibiotic_type	route	last_administration_day	days_since_last_admin	antibiotic_new	blood_culture_day	drug_in_bcx_window
1	1	2	ciprofloxacin	IV	NA	NA	1	3	1
2	1	4	ciprofloxacin	IV	2	2	0	3	1
3	1	6	ciprofloxacin	IV	4	2	0	3	0
4	1	7	doxycycline	IV	NA	NA	1	3	0
5	1	9	doxycycline	IV	7	2	0	3	0
6	1	15	penicillin	IV	NA	NA	1	3	0
7	1	16	doxycycline	IV	9	7	1	3	0
8	1	18	ciprofloxacin	IV	6	12	1	3	0
9	1	15	penicillin	IV	NA	NA	1	13	1
10	1	16	doxycycline	IV	9	7	1	13	0

4.5 Data Filtering:

The project employs iterative filtering techniques to identify patients with suspected infections. By checking for specific criteria and progressively filtering out irrelevant rows, the project narrows down to a group of patients who meet the requirements.

This is done to view only those antibiotics that were given through the route “I.V.” and neglect rest.

	patient_id	day_given	antibiotic_type	route	last_administration_day	days_since_last_admin	antibiotic_new	blood_culture_day	drug_in_bcx_window	any_iv_in_bcx_window
1	1	2	ciprofloxacin	IV	NA	NA	1	3	1	1
2	1	4	ciprofloxacin	IV	2	2	0	3	1	1
3	1	6	ciprofloxacin	IV	4	2	0	3	0	1
4	1	7	doxycycline	IV	NA	NA	1	3	0	1
5	1	9	doxycycline	IV	7	2	0	3	0	1
6	1	15	penicillin	IV	NA	NA	1	3	0	1
7	1	16	doxycycline	IV	9	7	1	3	0	1
8	1	18	ciprofloxacin	IV	6	12	1	3	0	1
9	1	15	penicillin	IV	NA	NA	1	13	1	1
10	1	16	doxycycline	IV	9	7	1	13	0	1

4.5.1 Step 1:

Also created a new variable in the merged dataset to find the first day of a new antibiotic sequence and remove data that is before the first qualifying day i.e. day_of_first_new_abx_in_window.

	antibiotic_type	route	last_administration_day	days_since_last_admin	antibiotic_new	blood_culture_day	drug_in_bcx_window	any_iv_in_bcx_window	day_of_first_new_abx_in_window
2	ciprofloxacin	IV	NA	NA	1	3	1	1	2
4	ciprofloxacin	IV	2	2	0	3	1	1	2
6	ciprofloxacin	IV	4	2	0	3	0	1	2
7	doxycycline	IV	NA	NA	1	3	0	1	2
9	doxycycline	IV	7	2	0	3	0	1	2
15	penicillin	IV	NA	NA	1	3	0	1	2
16	doxycycline	IV	9	7	1	3	0	1	2
18	ciprofloxacin	IV	6	12	1	3	0	1	2
15	penicillin	IV	NA	NA	1	13	1	1	15
16	doxycycline	IV	9	7	1	13	0	1	15

4.5.2 Step 2:

A new data frame is created which contains patient_id, blood_culture_day, and day_given from merge_dt.

	patient_id	blood_culture_day	day_given
1	1	3	2
2	1	3	4
3	1	3	6
4	1	3	7
5	1	3	9
6	1	3	15
7	1	3	16
8	1	3	18
9	8	2	1
10	8	2	2

4.6 Efficient Data Processing:

The use of data.table's special symbols like.SD and . N allows for efficient data processing, aggregations, and calculations. These symbols reduce the need for complex loops and contribute to better performance.

Hence a new variable is created in the new data frame that counts several antibiotic days. The blood_culture day with a count less than

or equal to four is filtered out.

	patient_id	blood_culture_day	day_given	num_antibiotic_days
1	1	3	2	8
2	1	3	4	8
3	1	3	6	8
4	1	3	7	8
5	1	3	9	8
6	1	3	15	8
7	1	3	16	8
8	1	3	18	8
9	8	2	1	6
10	8	2	2	6

Therefore another data frame called first_four_rows extracts the first four days of each culture using.SD function and four_in_seq variable are added to the dataset to depict antibiotic sequence with no skips of more than one day.

	patient_id	blood_culture_day	day_given	num_antibiotic_days	four_in_seq
1	1	3	2	8	1
2	1	3	4	8	1
3	1	3	6	8	1
4	1	3	7	8	1
5	8	2	1	6	0
6	8	2	2	6	0
7	8	2	3	6	0
8	8	2	6	6	0
9	23	3	1	13	1
10	23	3	3	13	1

4.7 Project Structure:

The project is well-structured into individual tasks (Task1, Task2, etc.), making it easier to manage and understand the workflow. Each task focuses on a specific aspect of the analysis, ensuring a step-by-step approach to solving the problem.

4.8 Interpretation of Results:

The final output of the project is a dataset containing patients with suspected infections. However, the interpretation of these results requires further analysis to get the best insights.

4.8.1 Step 1:

Another data frame is introduced called suspected_infection which includes only those patients whose four_in_seq is exactly equal to one. A new variable infection is included that is initialized by 1.

	patient_id	infection
1	1	1
2	23	1
3	64	1
4	76	1
5	164	1
6	176	1
7	200	1
8	204	1
9	206	1
10	213	1

4.8.2 Step 2:

Another dataset i.e. the all_patients dataset is included and merged with the suspected_infection dataset.

4.8.3 Step 3:

The new data set is cleaned and all the NA values are changed to 0.

	patient_id	infection
1	1	1
2	5	0
3	8	0
4	9	0
5	12	0
6	16	0
7	19	0
8	23	1
9	25	0
10	39	0

V. CONCLUSION

The "Classify Suspected Infection" project aimed to identify hospital patients with severe infections using medical record data. The project successfully leveraged the data. table package in R to perform various data manipulations and analyses to achieve its objectives.

According to the dataset provided, several data filtering, data cleaning, and data manipulation techniques were used to generate insights. Some statistical calculations like “mean” was used that helped us conclude that 14.94% of patients meet the criteria of presumed infection.

Overall, the "Classify Suspected Infection" project showcases the value of data analysis and the data. table package in healthcare research. The project's results can guide healthcare professionals in identifying patients at risk of severe infections, ultimately contributing to improved patient care and safety in hospital settings.

VI. FUTURE SCOPE

The "Classify Suspected Infection" project holds promising opportunities for future research and application. By utilizing advanced analytics, real-time monitoring, and integrating clinical data, it can enhance infection management, improve patient outcomes, and advance sepsis research.

Expanding the analysis to include data from multiple hospitals or healthcare facilities could increase the dataset's size and diversity. Analyzing data from different locations may reveal regional variations in infection patterns and treatment practices.

This project has exciting possibilities for the future. It can help us improve how we identify severe infections in patients. By using smarter ways to analyze medical data and real-time monitoring, doctors can detect infections early and treat patients more effectively. This project can lead to better patient care and potentially save lives. In the future, this research could be expanded to include more hospital and patient information, helping us understand infections better and find ways to prevent them. Ultimately, it has the potential to significantly impact healthcare and improve outcomes for patients with severe infections.

VII. CODE

```
#--------------------TASK 1-----------------------
install.packages("data.table")
#installed data.table package

library(data.table)
#library of data.table i.e where the package
resides

antibioticDT <- fread("C:\\Users\\Devashish
Chadha\\OneDrive\\Desktop\\Data anlytics by
Google\\Project\\project.data\\Classify Suspected
Infection in Patients\\datasets\\antibioticDT.csv")
#made a data frame "antibiotics_df" and using fread
function imported the data set

setorder(x = antibioticDT,patient_id,
antibiotic_type, day_given)
#sorted on the basis of patient_id,antibiotic_type
and then day_given

antibioticDT[1:40]
#Only included first 40 rows of the data set
```

```
#--------------------TASK 2-----------------------

antibioticDT[,last_administration_day :=
shift(day_given,n=1,fill = NA,type= "lag"),by =
.(patient_id,antibiotic_type)]
#calculates the last administration day of
antibiotics for each patient and antibiotic type by
creating a new column last_administration_day

antibioticDT[,days_since_last_admin := day_given -
last_administration_day]
#Calculate the number of days since the antibiotic
was administered to a patient

antibioticDT[,antibiotic_new := 1]
#created a new variable and initialized with 1

antibioticDT[days_since_last_admin <=
2,antibiotic_new := 0]
#one or two days since the last antibiotic was
given + initialized antibiotics_new to 0

antibioticDT[1:40]
#Only included first 40 rows of the data set
```

```
#--------------------TASK 3-----------------------
blood_cultureDT <- fread("C:\\Users\\Devashish
Chadha\\OneDrive\\Desktop\\Data anlytics by
Google\\Project\\project.data\\Classify Suspected
Infection in
Patients\\datasets\\blood_cultureDT.csv")
#imported blood culture data

blood_cultureDT[1:30]
#print first 30 rows

#--------------------TASK 4-----------------------
merged_df <- merge(antibioticDT,blood_cultureDT,by
= "patient_id",all = FALSE)
#merged two data sets having on the basis of common
patient id

setorder(merged_df,patient_id, blood_culture_day,
day_given, antibiotic_type)
#sorted merged data set on the basis of patient_id,
blood_culture_day, day_given, and antibiotic_type

merged_df[1:30]
#print merged data
```

```
#--------------------TASK 5-----------------------

merged_df[,drug_in_bcx_window :=
as.numeric((day_given - blood_culture_day) <= 2 &
(day_given - blood_culture_day) >= -2)]
#created a variable drug_in_bcx_window that checks
if the difference of day_given and
blood_culture_day is within 2 days

#--------------------TASK 6-----------------------

merged_df[,any_iv_in_bcx_window :=
as.numeric(any(route == "IV" & drug_in_bcx_window
== 1)),by = .(patient_id,blood_culture_day)]
#indicates whether antibiotic was given through
route "IV" or not in given day i.e
blood_culture_day

merged_df <- merged_df[any_iv_in_bcx_window == 1]
#excluded rows that do not include route "IV"
```

```
#--------------------TASK 7-----------------------

merged_df[,day_of_first_new_abx_in_window :=
day_given[antibiotic_new == 1 & drug_in_bcx_window
== 1][1],by = .(patient_id, blood_culture_day)]
#Created a new variable to find the first day of a
new antibiotic sequence

merged_df <- merged_df[day_given >=
day_of_first_new_abx_in_window]
#Removed data that is before first qualifying day

#--------------------TASK 8-----------------------

new_dt <-
merged_df[,.(patient_id,blood_culture_day,day_given
)]
#created a new data table which contains
patient_id, blood_culture_day,and day_given from
merge_dt

new_dt <- unique(new_dt)
#removed duplicate rows

new_dt[1:5]
#prints first 5 values
```

```
#--------------------TASK 9-----------------------
new_dt[,num_antibiotic_days := .N, by =
.(patient_id,blood_culture_day)]
#counts number of antibiotic days

new_dt <- new_dt[num_antibiotic_days >= 4]
#removed blood_culture_day with count less than
equal to 4

first_four_rows <- new_dt[, .SD[1:4], by =
.(patient_id,blood_culture_day)]
#prints first four days of each blood culture

first_four_rows[1:5]

#--------------------TASK 10-----------------------
first_four_rows[,four_in_seq :=
as.numeric(max(diff(day_given)) < 3), by =
.(patient_id,blood_culture_day)]
#antibiotic sequence no skips of more than one day
```

```
#--------------------TASK 11----------------------

suspected_infection <-
first_four_rows[four_in_seq == 1]
#rows with four_in_seq = 1

suspected_infection <-
suspected_infection[,.(patient_id)]
#Retained patient_id column

suspected_infection <-
unique(suspected_infection)
#removed duplicate values

suspected_infection <-
suspected_infection[,infection  := 1]
#inserted a new column "infection"
initializing it with 1

suspected_infection[1:5]
#prints first 5 values
```

```
#--------------------TASK 12-----------------------

all_patientsDT  <- fread("C:\\Users\\Devashish
Chadha\\OneDrive\\Desktop\\Data anlytics by
Google\\Project\\project.data\\Classify Suspected
Infection in Patients\\datasets\\all_patients.csv")
#read "all_patients" data set

all_patientsDT <- merge(all_patientsDT,
suspected_infection, all = TRUE)
#merged suspected_infections data frame and
all_patients data frame

all_patientsDT <- all_patientsDT[is.na(infection),
infection := 0]
#replaced NA in infection to 0

all_patientsDT[1:10]
#prints first 10 rows

percentage <- 100 * all_patientsDT[,
mean(infection)]
#created a value which tells the percentage of
patients who met the criteria for presumed
infection
```

VI. REFERENCES

Data Collection

The following websites have been referred to obtain the input data:

a. https://www.who.int/news-room/fact-sheets/detail/sepsis

b. https://www.r-project.org/about.html

c. https://drive.google.com/file/d/162KBAQEUU6b84LIgj8MU3a7E6Yrel_bg/view (Datasets)

d. https://www.geeksforgeeks.org/introduction-to-r-studio/

e. https://www.projectpro.io/data-science-in-r-programming-tutorial/r-data-table-tutorial

Programming References

The following websites have been referred for R coding and data. table package:

a. https://www.r-project.org/other-docs.html

b. https://search.r-project.org/CRAN/refmans/binhf/html/shift.html

c. https://www.tutorialspoint.com/r/r_mean_median_mode.htm