COVID 19 Data Report analysis for India

2024-08-05

Dataset Description

This extensive dataset acts as a powerful tool for grasping the worldwide situation of COVID-19. Meticulously compiled and kept up-to-date by the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE), it offers daily summaries over time, meticulously tracking confirmed cases, fatalities, and recoveries since January 21st, 2020.

The data originates from a variety of trustworthy sources, including the World Health Organization (WHO), the Los Angeles Times, and QQ News. This collaborative effort ensured a thorough picture of the pandemic's development across the globe.

It's important to note, however, that data collection by the Johns Hopkins Coronavirus Resource Center stopped on March 10th, 2023. This cessation signifies a significant change in how we track and monitor the pandemic.

For those seeking to explore this dataset further, a wealth of information is readily available on the official Johns Hopkins Github repository: link to repository https://github.com/CSSEGISandData/COVID-19.

Step 0: Import Packages

```
library(tidyverse)
library(forecast)
#tidyverse provides a suite of data manipulation tools.
#forecast is used for time series analysis and forecasting.
```

Step 1: Import the Data

- Copy the URL of the csv file.

```
# Define the base URL for COVID-19 data
url_in = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covi


# Specify filenames for confirmed and death cases
file_names = c("time_series_covid19_confirmed_global.csv", "time_series_covid19_deaths_global.csv")

# Combine URL and filenames to create download links
urls <- str_c(url_in, file_names)

# Print the download links for reference
cat("Download URLs:\n", urls, "\n")
```

```
## Download URLs:
##  https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_ti
```

Import Description

- The code defines the base URL for the COVID-19 data repository.
- It specifies filenames for both confirmed cases and death cases.
- str_c() is used to concatenate the URL and filenames, creating download links.
- read_csv() reads the confirmed cases data from the first URL.
- Use `read_csv()` to read in the data.

```
# Read confirmed cases data here
global_cases <- read_csv(urls[1])
```

```
# Read death cases data here
global_deaths <- read_csv(urls[2])
```

```
## # A tibble: 6 x 1,147
##   `Province/State` `Country/Region`   Lat  Long `1/22/20` `1/23/20` `1/24/20`
##   <chr>            <chr>            <dbl> <dbl>     <dbl>     <dbl>     <dbl>
## 1 <NA>             Afghanistan       33.9 67.7          0         0         0
## 2 <NA>             Albania           41.2 20.2          0         0         0
## 3 <NA>             Algeria           28.0  1.66         0         0         0
## 4 <NA>             Andorra           42.5  1.52         0         0         0
## 5 <NA>             Angola           -11.2 17.9          0         0         0
## 6 <NA>             Antarctica       -71.9 23.3          0         0         0
## # i 1,140 more variables: `1/25/20` <dbl>, `1/26/20` <dbl>, `1/27/20` <dbl>,
## #   `1/28/20` <dbl>, `1/29/20` <dbl>, `1/30/20` <dbl>, `1/31/20` <dbl>,
## #   `2/1/20` <dbl>, `2/2/20` <dbl>, `2/3/20` <dbl>, `2/4/20` <dbl>,
## #   `2/5/20` <dbl>, `2/6/20` <dbl>, `2/7/20` <dbl>, `2/8/20` <dbl>,
## #   `2/9/20` <dbl>, `2/10/20` <dbl>, `2/11/20` <dbl>, `2/12/20` <dbl>,
## #   `2/13/20` <dbl>, `2/14/20` <dbl>, `2/15/20` <dbl>, `2/16/20` <dbl>,
## #   `2/17/20` <dbl>, `2/18/20` <dbl>, `2/19/20` <dbl>, `2/20/20` <dbl>, ...
```

```
## # A tibble: 6 x 1,147
##   `Province/State` `Country/Region`   Lat  Long `1/22/20` `1/23/20` `1/24/20`
##   <chr>            <chr>            <dbl> <dbl>     <dbl>     <dbl>     <dbl>
## 1 <NA>             Afghanistan       33.9 67.7          0         0         0
## 2 <NA>             Albania           41.2 20.2          0         0         0
## 3 <NA>             Algeria           28.0  1.66         0         0         0
## 4 <NA>             Andorra           42.5  1.52         0         0         0
## 5 <NA>             Angola           -11.2 17.9          0         0         0
## 6 <NA>             Antarctica       -71.9 23.3          0         0         0
## # i 1,140 more variables: `1/25/20` <dbl>, `1/26/20` <dbl>, `1/27/20` <dbl>,
## #   `1/28/20` <dbl>, `1/29/20` <dbl>, `1/30/20` <dbl>, `1/31/20` <dbl>,
## #   `2/1/20` <dbl>, `2/2/20` <dbl>, `2/3/20` <dbl>, `2/4/20` <dbl>,
## #   `2/5/20` <dbl>, `2/6/20` <dbl>, `2/7/20` <dbl>, `2/8/20` <dbl>,
## #   `2/9/20` <dbl>, `2/10/20` <dbl>, `2/11/20` <dbl>, `2/12/20` <dbl>,
## #   `2/13/20` <dbl>, `2/14/20` <dbl>, `2/15/20` <dbl>, `2/16/20` <dbl>,
## #   `2/17/20` <dbl>, `2/18/20` <dbl>, `2/19/20` <dbl>, `2/20/20` <dbl>, ...
```

Step 2: Tidy and Transform the Data

1. Tidy the columns

- Put each variable (**date**, **cases**, and **deaths**) in their own column.
- Remove columns: **Lat** and **Long**.
- Rename columns: **Province/State** and **Country/Region**.
- Convert column **date** to date object.

```
# Use `pivot_longer()` to make each date on a separate row
tidy_cases = global_cases %>%
  pivot_longer(cols = -c(`Province/State`, `Country/Region`, Lat, Long), names_to = "date",
               values_to = "cases")

tidy_deaths = global_deaths %>%
  pivot_longer(cols = -c(`Province/State`, `Country/Region`, Lat, Long), names_to = "date",
               values_to = "deaths")
```

```
# Combine confirmed and death cases data using full_join()
global_tidy <- tidy_cases %>%
  full_join(tidy_deaths) %>%
  select(-c(Lat, Long)) %>%
  rename(Country_Region = `Country/Region`, Province_State = `Province/State`) %>%
  mutate(date = mdy(date))

# View the first few rows of the tidied data
head(global_tidy)
```

```
## # A tibble: 6 x 5
##   Province_State Country_Region date       cases deaths
##   <chr>          <chr>          <date>     <dbl>  <dbl>
## 1 <NA>           Afghanistan    2020-01-22     0      0
## 2 <NA>           Afghanistan    2020-01-23     0      0
## 3 <NA>           Afghanistan    2020-01-24     0      0
## 4 <NA>           Afghanistan    2020-01-25     0      0
## 5 <NA>           Afghanistan    2020-01-26     0      0
## 6 <NA>           Afghanistan    2020-01-27     0      0
```

2. Tidy the rows

- Filter the rows of **Country_Region** of India*.

```
# Filter data for India
india_tidy <- global_tidy %>%
  filter(Country_Region == "India") %>%
  select(-Province_State)

# Summarize the tidied data for India
summary(india_tidy)
```

```
##  Country_Region          date                 cases              deaths
##  Length:1143        Min.   :2020-01-22   Min.   :       0   Min.   :     0
##  Class :character   1st Qu.:2020-11-02   1st Qu.: 8290750   1st Qu.:123354
##  Mode  :character   Median :2021-08-15   Median :32225513   Median :431642
##                     Mean   :2021-08-15   Mean   :25486544   Mean   :319266
##                     3rd Qu.:2022-05-27   3rd Qu.:43151629   3rd Qu.:524579
##                     Max.   :2023-03-09   Max.   :44690738   Max.   :530779
```

Step 3: Add Visualizations and Analysis

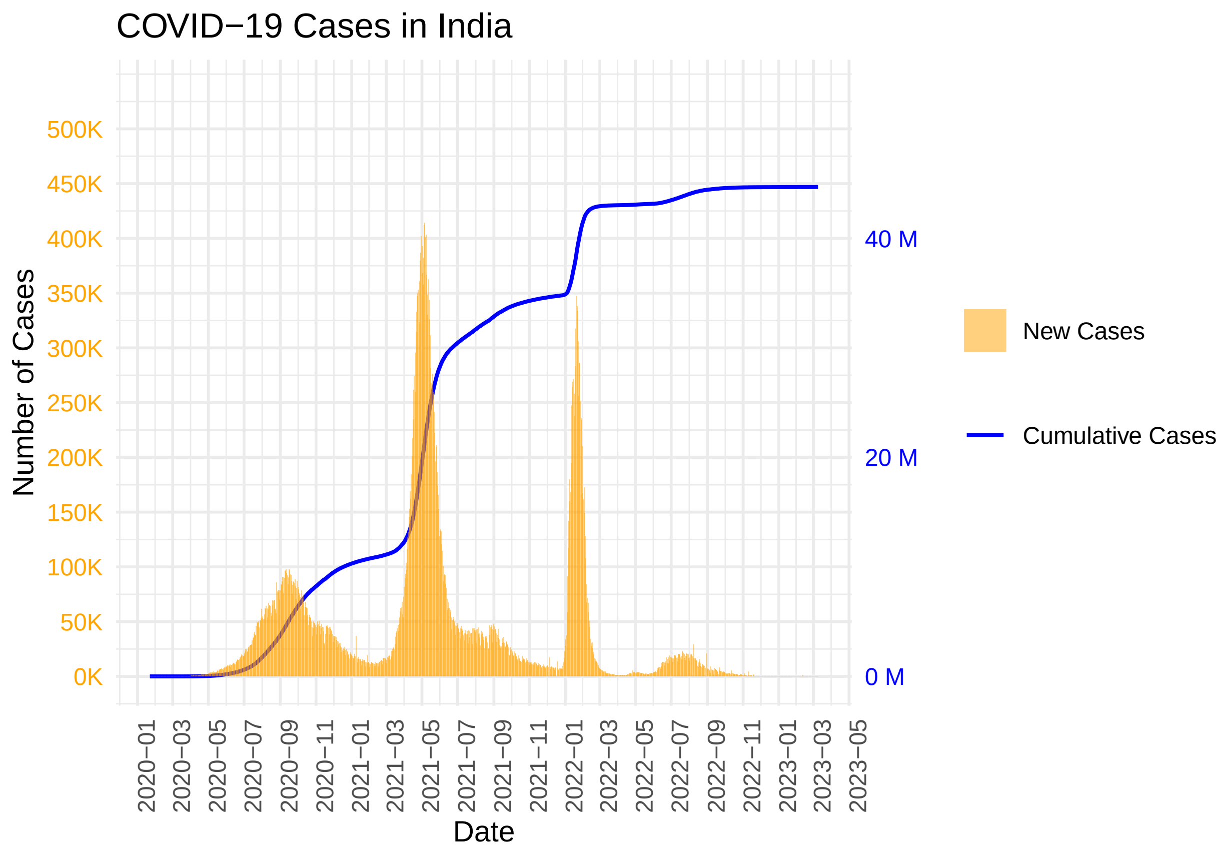
Question 1: What are the trends for daily cumulative confirmed cases and new confirmed cases of COVID-19 in India?

- Since the spread of the COVID-19 pandemic in January 2020, India has seen a significant surge in confirmed cases during April-May 2021.
- After the initial peak, there were subsequent waves with higher cases in January 2022 and July 2022.

```r
# Calculate new cases
india_tidy <- india_tidy %>%
  mutate(new_cases = cases - lag(cases)) %>%
  mutate(new_cases = ifelse(is.na(new_cases) | new_cases < 0, 0, new_cases))

# Convert cases and new cases to thousands for better visualization
india_tidy$cases_100k <- india_tidy$cases / 100000
india_tidy$new_cases_k <- india_tidy$new_cases / 1000

# Create the plot
ggplot(india_tidy, aes(x = date)) +
  geom_line(aes(y = cases_100k, color = "Cumulative Cases"), linewidth = 0.7) +
  geom_bar(aes(y = new_cases_k, fill = "New Cases"), stat = "identity", alpha = 0.5) +
  labs(x = "Date", y = "Number of Cases") +
  scale_color_manual(values = c("Cumulative Cases" = "blue")) +
  scale_fill_manual(values = c("New Cases" = "orange")) +
  ggtitle("COVID-19 Cases in India") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        axis.text.y.right = element_text(color = "blue"),
        axis.text.y.left = element_text(color = "orange"),
        axis.title.y.left = element_text(color = "black")) +
  scale_y_continuous(
    sec.axis = sec_axis(~.*100000, labels = scales::unit_format(unit = "M", scale = 1e-6)),
    limits = c(0, max(india_tidy$cases_100k, na.rm = TRUE) * 1.2),  # Adjust limits based on data
    breaks = seq(0, max(india_tidy$cases_100k, na.rm = TRUE) * 1.2, by = 50), labels = function(x) paste
  ) +
  scale_x_date(date_labels = "%Y-%m", date_breaks = "2 month") +
  guides(color = guide_legend(title = NULL), fill = guide_legend(title = NULL))
```



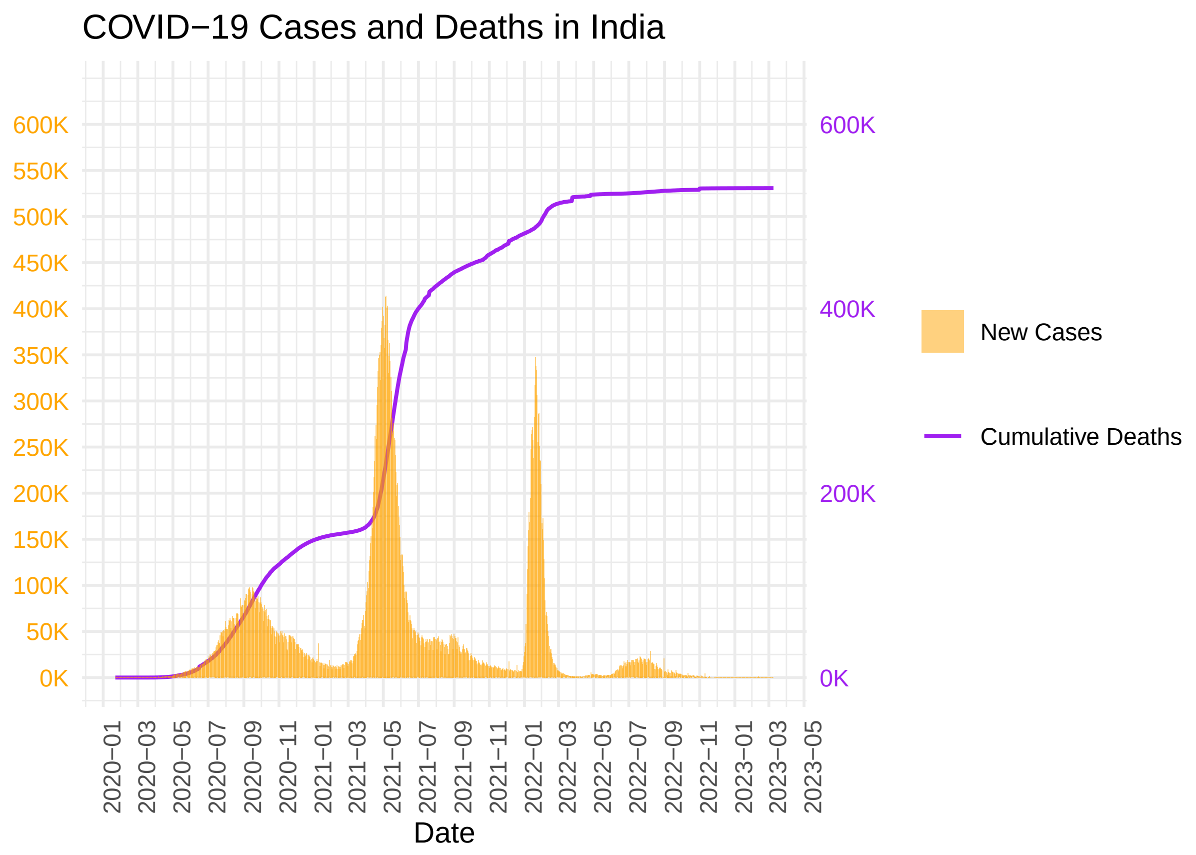
The code calculates new cases for the India, handles potential NA or negative values, and converts cases and new cases to thousands for better visualization. It creates a line chart for cumulative cases and a bar chart for new cases, with appropriate labels and formatting.

Question 2: What are the trends for daily cumulative deaths and new confirmed cases of COVID-19 in India?

- The number of deaths in India has also shown a significant increase during the peak periods of confirmed cases.

```r
# Convert cumulative deaths to thousands
india_tidy$deaths_k <- india_tidy$deaths / 1000

# Create the plot
ggplot(india_tidy, aes(x = date)) +
  geom_line(aes(y = deaths_k, color = "Cumulative Deaths"), linewidth = 0.7) +
  geom_bar(aes(y = new_cases_k, fill = "New Cases"), stat = "identity", alpha = 0.5) +
  labs(x = "Date", y = " ") +
  scale_color_manual(values = c("Cumulative Deaths" = "purple")) +
  scale_fill_manual(values = c("New Cases" = "orange")) +
  ggtitle("COVID-19 Cases and Deaths in India") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        axis.text.y.right = element_text(color = "purple"),
        axis.text.y.left = element_text(color = "orange")) +
  scale_y_continuous(
    sec.axis = sec_axis(~., labels = function(x) paste0(x, "K")),
    limits = c(0, max(india_tidy$deaths_k, na.rm = TRUE) * 1.2),  # Adjust limits based on data
    breaks = seq(0, max(india_tidy$deaths_k, na.rm = TRUE) * 1.2, by = 50), labels = function(x) paste0
  ) +
  scale_x_date(date_labels = "%Y-%m", date_breaks = "2 month") +
  guides(color = guide_legend(title = NULL), fill = guide_legend(title = NULL))
```



The code converts cumulative deaths to thousands for better visualization. It creates a line chart for cumulative deaths and a bar chart for new cases, with appropriate labels and formatting.

Question 3: Can we predict the future number of confirmed cases in India?

- Purpose: Predict the future number of COVID-19 confirmed cases in India for the upcoming year based on the data collected by JHU CSSE.
- Methods: Use an ARIMA model to model and forecast. - Use auto.arima() to build a time series model. - Use forecast() to predict future data.

```r
# Use `ts()` convert data into a time series object
ts_cases = ts(india_tidy$cases)

# ARIMA model
arima_model = auto.arima(ts_cases)
```

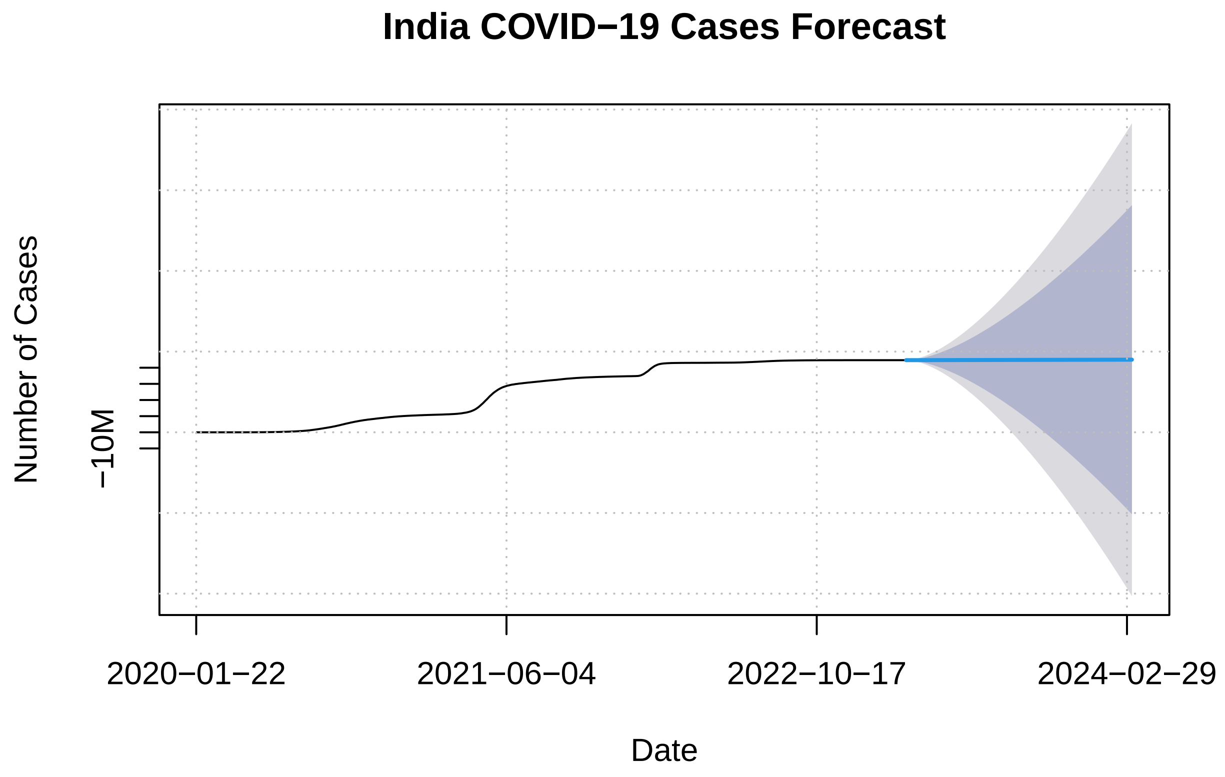
```r
# Make predictions using the established ARIMA model
future_forecast = forecast(arima_model, h = 365)

# Create a chart
plot(future_forecast, main = "India COVID-19 Cases Forecast", yaxt = "n", xaxt = "n")
grid(lty = "dotted", col = "gray")

# Draw the y-axis labels
y_labels = c(-10, 0, 10, 20, 30, 40) * 1e6
axis(2, at = y_labels, labels = paste0(y_labels / 1e6, "M"))

# Draw the x-axis labels
x_labels = c(0, 500, 1000, 1500)
x_labels_dates = c(india_tidy$date[1], india_tidy$date[1] + 499, india_tidy$date[1] + 999,india_tidy$da
axis(1, at = x_labels, labels = paste0(x_labels_dates))

# Add text to x-axis and y-axis
mtext("Date", side = 1, line = 3)
mtext("Number of Cases", side = 2, line = 3)
```



- The code creates a time series object (india_ts_cases) for the total cases in the India, specifying the start date and frequency.
- An ARIMA model is fitted to the time series data using auto.arima().
- The forecast() function generates predictions for the next 30 days based on the fitted model.
- The plot() function visualizes the forecast.

Step 4: Add Bias Identification

Bias Identification for the India COVID-19 Analysis

1. Personal Bias

- Before analysis: Given the extensive media coverage of India's COVID-19 challenges, there might be a tendency to focus on the peak periods while overlooking other trends.
- After analysis: Overreliance on specific data points or visualizations could lead to a narrow interpretation, neglecting potential alternative explanations for observed patterns.

2. Other Bias

- Reporting Bias: India's vast geographical area and diverse healthcare infrastructure might introduce variations in data collection and reporting practices across different regions. This could affect data consistency and comparability.
- Temporal Bias: India's experience with COVID-19 has been marked by distinct phases, including lockdowns, reopenings, and new variants. Analyzing data without considering these temporal factors might lead to misleading conclusions.
- Policy Bias: The impact of government policies, such as lockdown measures, vaccination campaigns, and testing strategies, can significantly influence the data. Overlooking these factors could bias the analysis.

Conclusion

This analysis provides a comprehensive overview of COVID-19 trends in India, including visualizations, time series modeling, and forecasting. The results can be used to understand the past, assess the present, and make predictions about the future trajectory of the pandemic. Further analysis and refinement of the models can provide more accurate and informative insights.

Note: This analysis is based on the provided data and assumptions. For a more comprehensive and accurate understanding of the COVID-19 situation in India, additional data sources, statistical methods, and domain expertise should be considered.