# COVID - 19 ( India ) Data Analysis And Visualization

This notebook uses data analysis and visualization to analyze the effects of the ongoing COVID-19 pandemic in India , and create visualizations for important observation made during the analysis.

Language : - Python 3

Dataset : COVID-19 in India

Libraries :

* Numpy
* Pandas
* Seaborn
* Matplotlib

# Importing Libraries

In [1]:
```
# import Libraries for data analysis

import numpy as np
import pandas as pd

# Import Libraries for data visualization

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set_style(style ='whitegrid', rc = {'xtrick.bottom': True, 'ytick.left': True}
```

# Reading , exploring and cleaning the data

In [2]:
```
# read data from the dataset into dataframes
age_group_details = pd.read_csv("E:\COVID19 DATA\AgeGroupDetails COVID19 INDIA.csv'
covid_19_india = pd.read_csv("E:\COVID19 DATA\covid_19_india COVID19 INDIA.csv")
hospital_beds_india = pd.read_csv("E:\COVID19 DATA\HospitalBedsIndiaCOVID19 INDIA.c
icmr_testing_details = pd.read_csv("E:\COVID19 DATA\ICMRTestingDetails COVID19 IND
individual_details = pd.read_csv("E:\COVID19 DATA\IndividualDetails COVID19 INDIA.c
population_india_census_2011 = pd.read_csv("E:\COVID19 DATA\population_india_censu
statewise_testing_details = pd.read_csv("E:\COVID19 DATA\StatewiseTestingDetails C
districtwise_testing_details = pd.read_csv("E:\COVID19 DATA\district_level_latest c
covid_19_details = pd.read_csv("E:\COVID19 DATA\covid_19_india COVID19 INDIA.csv")
```

Exploring each of the dataframes (below)

In [3]:
```
age_group_details.head()
```

Out[3]:

| | Sno | AgeGroup | TotalCases | Percentage |
|---|---|---|---|---|
| **0** | 1 | 0-9 | 22 | 3.18% |
| **1** | 2 | 10-19 | 27 | 3.90% |
| **2** | 3 | 20-29 | 172 | 24.86% |
| **3** | 4 | 30-39 | 146 | 21.10% |
| **4** | 5 | 40-49 | 112 | 16.18% |

In [4]:
```
covid_19_india.head()
```

Out[4]:

| | Sno | Date | Time | State/UnionTerritory | ConfirmedIndianNational | ConfirmedForeignNational |
|---|---|---|---|---|---|---|
| **0** | 1 | 30/01/20 | 6:00 PM | Kerala | 1 | 0 |
| **1** | 2 | 31/01/20 | 6:00 PM | Kerala | 1 | 0 |
| **2** | 3 | 01/02/20 | 6:00 PM | Kerala | 2 | 0 |
| **3** | 4 | 02/02/20 | 6:00 PM | Kerala | 3 | 0 |
| **4** | 5 | 03/02/20 | 6:00 PM | Kerala | 3 | 0 |

In [5]:
```
hospital_beds_india.head()
```

Out[5]:

| | Sno | State/UT | NumPrimaryHealthCenters_HMIS | NumCommunityHealthCenters_HMIS | NumSub |
|---|---|---|---|---|---|
| **0** | 1 | Andaman & Nicobar Islands | 27 | 4 | |
| **1** | 2 | Andhra Pradesh | 1417 | 198 | |
| **2** | 3 | Arunachal Pradesh | 122 | 62 | |
| **3** | 4 | Assam | 1007 | 166 | |
| **4** | 5 | Bihar | 2007 | 63 | |

In [6]:
```
icmr_testing_details.head()
```

Out[6]:

| | SNo | DateTime | TotalSamplesTested | TotalIndividualsTested | TotalPositiveCases |
|---|---|---|---|---|---|
| **0** | 1 | 13/03/20 18:00 | 6500.0 | 5900.0 | 78.0 |
| **1** | 2 | 18/03/20 18:00 | 13125.0 | 12235.0 | 150.0 |
| **2** | 3 | 19/03/20 18:00 | 14175.0 | 13285.0 | 182.0 |
| **3** | 4 | 20/03/20 18:00 | 15404.0 | 14514.0 | 236.0 |
| **4** | 5 | 21/03/20 18:00 | 16911.0 | 16021.0 | 315.0 |

In [7]:
```
individual_details.head()
```

Out[7]:

| | id | government_id | diagnosed_date | age | gender | detected_city | detected_district | detected_sta |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | KL-TS-P1 | 30/01/2020 | 20 | F | Thrissur | Thrissur | Kera |
| **1** | 1 | KL-AL-P1 | 02/02/2020 | NaN | NaN | Alappuzha | Alappuzha | Kera |
| **2** | 2 | KL-KS-P1 | 03/02/2020 | NaN | NaN | Kasaragod | Kasaragod | Kera |
| **3** | 3 | DL-P1 | 02/03/2020 | 45 | M | East Delhi (Mayur Vihar) | East Delhi | De |
| **4** | 4 | TS-P1 | 02/03/2020 | 24 | M | Hyderabad | Hyderabad | Telanga |

In [8]:
```
population_india_census_2011.head()
```

Out[8]:

| | Sno | State / Union Territory | Population | Rural population | Urban population | Area | |
|---|---|---|---|---|---|---|---|
| **0** | 1 | Uttar Pradesh | 199812341 | 155317278 | 44495063 | 240,928 km2 (93,023 sq mi) | 828/km2 (2, |
| **1** | 2 | Maharashtra | 112374333 | 61556074 | 50818259 | 307,713 km2 (118,809 sq mi) | 365/km2 (9 |
| **2** | 3 | Bihar | 104099452 | 92341436 | 11758016 | 94,163 km2 (36,357 sq mi) | 1,102/km2 (2,8 |
| **3** | 4 | West Bengal | 91276115 | 62183113 | 29093002 | 88,752 km2 (34,267 sq mi) | 1,029/km2 (2,( |
| **4** | 5 | Madhya Pradesh | 72626809 | 52557404 | 20069405 | 308,245 km2 (119,014 sq mi) | 236/km2 (( |

In [9]:
```
statewise_testing_details.head()
```

Out[9]:

| | Date | State | TotalSamples | Negative | Positive |
|---|---|---|---|---|---|
| **0** | 2020-04-17 | Andaman and Nicobar Islands | 1403.0 | 1210.0 | 12.0 |
| **1** | 2020-04-24 | Andaman and Nicobar Islands | 2679.0 | NaN | 27.0 |
| **2** | 2020-04-27 | Andaman and Nicobar Islands | 2848.0 | NaN | 33.0 |
| **3** | 2020-05-01 | Andaman and Nicobar Islands | 3754.0 | NaN | 33.0 |
| **4** | 2020-04-02 | Andhra Pradesh | 1800.0 | 1175.0 | 132.0 |

In [10]:
```
districtwise_testing_details.head()
```

Out[10]:

| | SlNo | State_Code | State | District_Key | District | Confirmed | Active | Recovered | Dece |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | UN | State Unassigned | UN_Unassigned | Unassigned | 0 | 0 | 0 | |
| **1** | 1 | AN | Andaman and Nicobar Islands | AN_Nicobars | Nicobars | 0 | 0 | 0 | |
| **2** | 2 | AN | Andaman and Nicobar Islands | AN_North and Middle Andaman | North and Middle Andaman | 1 | 0 | 1 | |
| **3** | 3 | AN | Andaman and Nicobar Islands | AN_South Andaman | South Andaman | 51 | 19 | 32 | |
| **4** | 0 | AP | Andhra Pradesh | AP_Foreign Evacuees | Foreign Evacuees | 434 | 0 | 434 | |

In [11]:
```
covid_19_details.head()
```

Out[11]:

| | Sno | Date | Time | State/UnionTerritory | ConfirmedIndianNational | ConfirmedForeignNational |
|---|---|---|---|---|---|---|
| **0** | 1 | 30/01/20 | 6:00 PM | Kerala | 1 | 0 |
| **1** | 2 | 31/01/20 | 6:00 PM | Kerala | 1 | 0 |
| **2** | 3 | 01/02/20 | 6:00 PM | Kerala | 2 | 0 |
| **3** | 4 | 02/02/20 | 6:00 PM | Kerala | 3 | 0 |
| **4** | 5 | 03/02/20 | 6:00 PM | Kerala | 3 | 0 |

In [12]:
```
# update the state name for Telangana
population_india_census_2011['State / Union Territory'] = population_india_census_

# update the district name for Ahmedabad
individual_details['detected_district'] = individual_details['detected_district'].
```

```
# udpate the city name for Ahmedabad
individual_details['detected_city'] = individual_details['detected_city'].apply(la
```

# Data Analysis And Visualization

1. Cumulative number of cases (categorised by current health status) grouped by date.

1.1 Creating a dataframe with number of cases (categorised by current health status) grouped by date [CUMULATIVE]

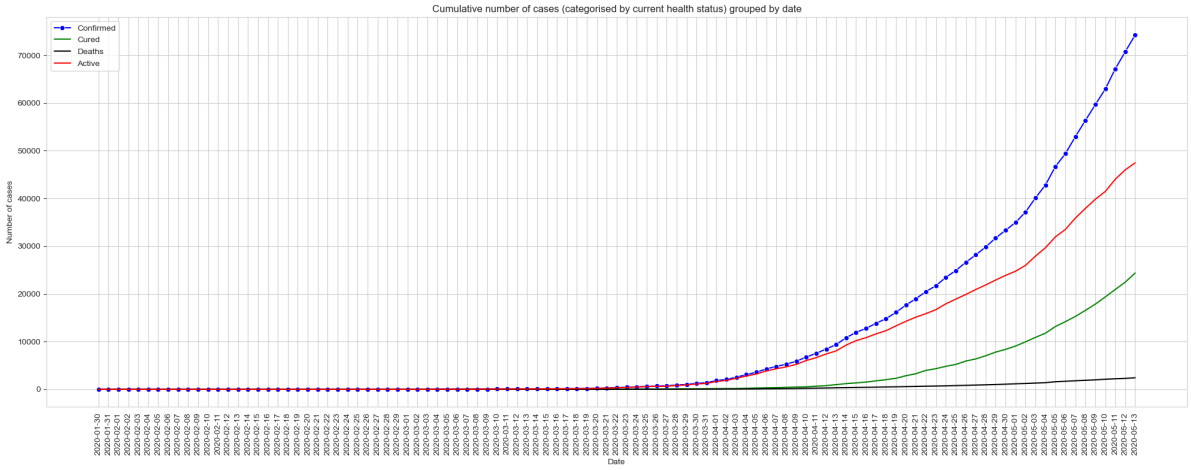
In [13]:
```
date_cumulative = covid_19_india.groupby('Date').sum()
date_cumulative.reset_index(inplace = True)

# change the date format to 'YYYY-MM-DD'
date_cumulative['Date'] = date_cumulative['Date'].apply(lambda date : '20' + '-'.j

# sort the rows by date (in ascending order)
date_cumulative.sort_values('Date', inplace = True)

# calculate the number of active cases
date_cumulative['Active'] = date_cumulative['Confirmed'] - (date_cumulative['Cured

date_cumulative = date_cumulative[['Date', 'Confirmed', 'Cured', 'Deaths', 'Active

# dataframe with number of cases (categorised by current health status) grouped by
date_cumulative.head()
```

Out[13]:

| | Date | Confirmed | Cured | Deaths | Active |
|---|---|---|---|---|---|
| **100** | 2020-01-30 | 1 | 0 | 0 | 1 |
| **103** | 2020-01-31 | 1 | 0 | 0 | 1 |
| **0** | 2020-02-01 | 2 | 0 | 0 | 2 |
| **4** | 2020-02-02 | 3 | 0 | 0 | 3 |
| **8** | 2020-02-03 | 3 | 0 | 0 | 3 |

# 1.2 Plot

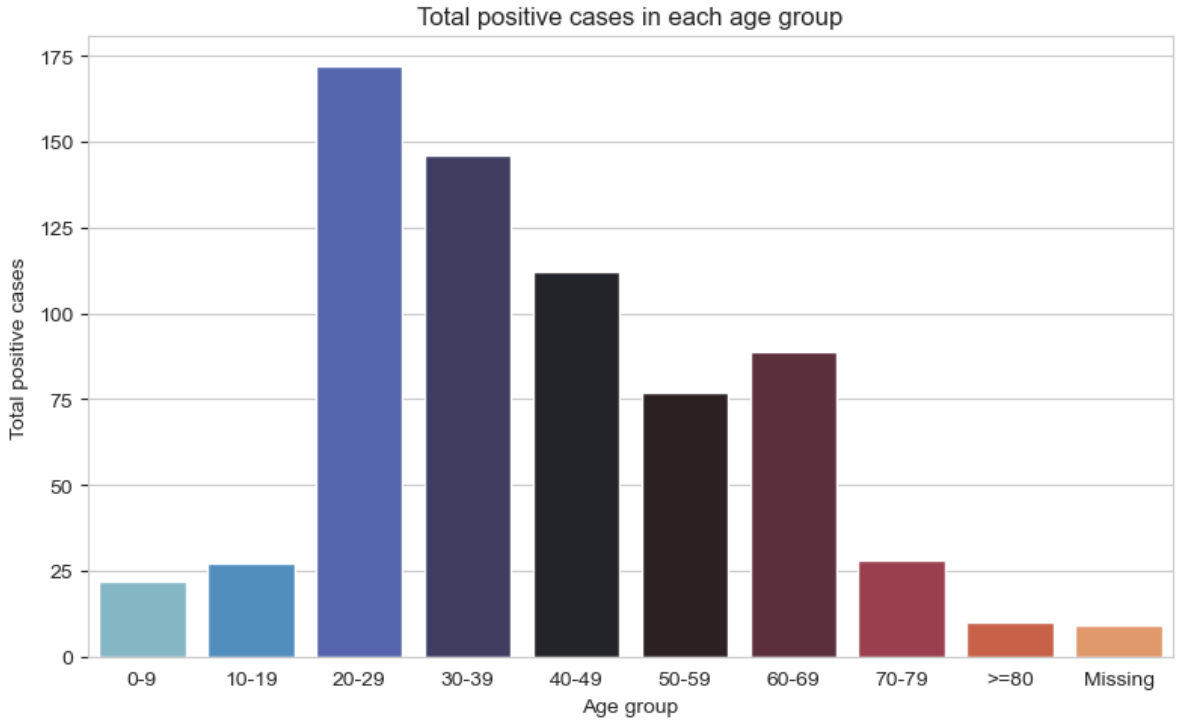
In [14]:
```
plt.figure(figsize=(20, 8), dpi = 100)
sns.lineplot(x = 'Date', y = 'Confirmed', data = date_cumulative, label = 'Confirm
sns.lineplot(x = 'Date', y = 'Cured', data = date_cumulative, label = 'Cured', col
sns.lineplot(x = 'Date', y = 'Deaths', data = date_cumulative, label = 'Deaths', c
sns.lineplot(x = 'Date', y = 'Active', data = date_cumulative, label = 'Active', c
plt.title('Cumulative number of cases (categorised by current health status) groupe
plt.ylabel('Number of cases')
plt.legend(loc = 0)
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```




# 2. Total Positive Cases In Each Age Group

2.1 Plot

In [15]:
```
plt.figure(figsize=(8, 5), dpi = 100)
sns.barplot(x = 'AgeGroup', y = 'TotalCases', data = age_group_details, palette =
plt.title('Total positive cases in each age group')
plt.xlabel('Age group')
plt.ylabel('Total positive cases')
plt.tight_layout()
plt.show()
```




# 3 Total cases in india each of its state and union territories

3.1 Creating a dataframe with important data from each state/union territory as well as india

In [16]:
```
temp_state = covid_19_india.groupby('State/UnionTerritory').max()
temp_state.reset_index(inplace = True)
temp_state['Active'] = temp_state['Confirmed'] - (temp_state['Cured'] + temp_state
temp_state = temp_state[['State/UnionTerritory', 'Confirmed', 'Cured', 'Deaths', '
temp_state.rename(columns = {'State/UnionTerritory': 'State / Union Territory'}, i
temp_state['State / Union Territory'] = temp_state['State / Union Territory'].appl

# calculate the number of cases (categorised by current health status) for India a
# index number 41 is chosen to avoid any loss of data (as of now, the total number
temp_state.loc[41] = ['India', temp_state['Confirmed'].sum(), temp_state['Cured'].

temp_state.head()
```

Out[16]:

| | State / Union Territory | Confirmed | Cured | Deaths | Active |
|---|---|---|---|---|---|
| **0** | Andaman and Nicobar Islands | 33 | 33 | 0 | 0 |
| **1** | Andhra Pradesh | 2090 | 1056 | 46 | 988 |
| **2** | Arunachal Pradesh | 1 | 1 | 0 | 0 |
| **3** | Assam | 65 | 39 | 2 | 24 |
| **4** | Bihar | 831 | 383 | 6 | 442 |

In [17]:
```
# calculate India's total area
total_area_india_km2 = population_india_census_2011['Area'].apply(lambda area : fl

temp_population = population_india_census_2011[['State / Union Territory', 'Popula
temp_population['Population density (per km2)'] = temp_population['Density'].apply
temp_population.drop(columns = ['Density'], inplace = True)

# calculate India's total population
total_population_india = temp_population['Population'].sum()

# calculate India's total population density
density_india = total_population_india / total_area_india_km2

temp_population.loc[41] = ['India', total_population_india, density_india]
temp_population.head()
```

```
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\3334464457.py:5: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  temp_population['Population density (per km2)'] = temp_population['Density'].app
ly(lambda density : float(density.split('/')[0].replace(',', '')))
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\3334464457.py:6: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  temp_population.drop(columns = ['Density'], inplace = True)
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\3334464457.py:14: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  temp_population.loc[41] = ['India', total_population_india, density_india]
```

Out[17]:

| | State / Union Territory | Population | Population density (per km2) |
|---|---|---|---|
| **0** | Uttar Pradesh | 199812341 | 828.0 |
| **1** | Maharashtra | 112374333 | 365.0 |
| **2** | Bihar | 104099452 | 1102.0 |
| **3** | West Bengal | 91276115 | 1029.0 |
| **4** | Madhya Pradesh | 72626809 | 236.0 |

In [18]:

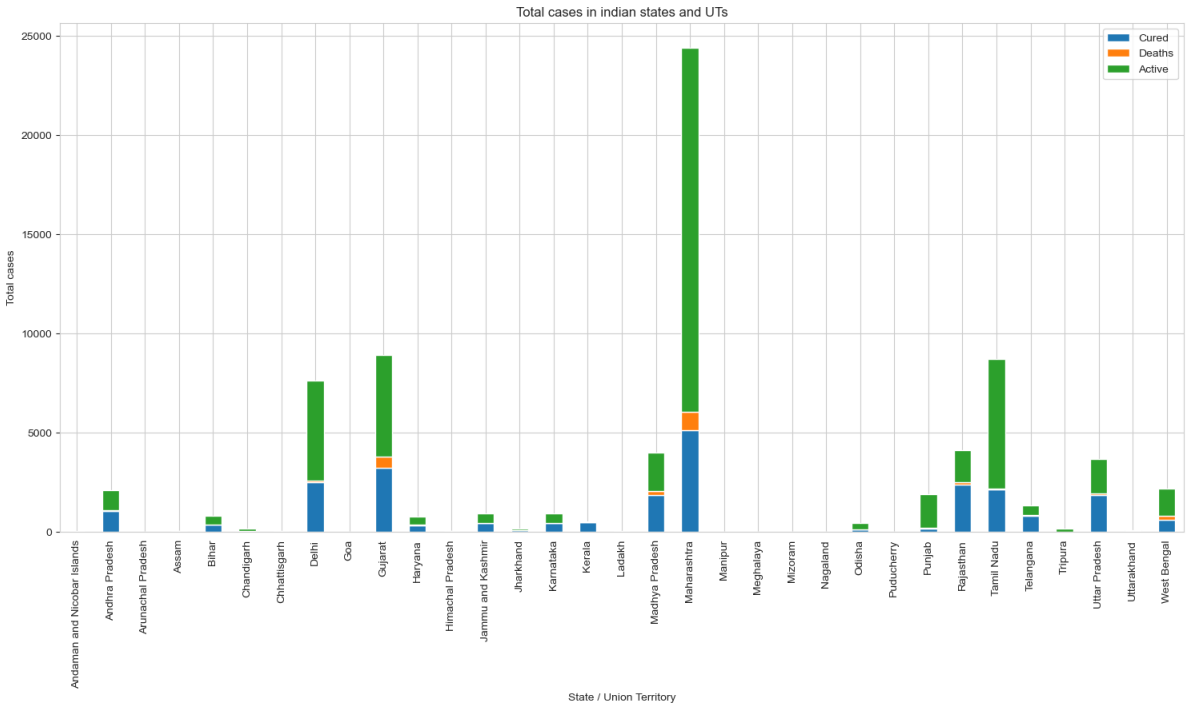
```
statewise_data = pd.merge(left = temp_state, right = temp_population, on = 'State

# select India's data from the merged dataframe and storing it as a series in a var
india_data = statewise_data.iloc[-1]

# drop India's data stored in the last row of the dataframe
statewise_data.drop(statewise_data.tail(1).index, inplace = True)

# store India's data in a row with index number 41 in the dataframe
statewise_data.loc[41] = india_data

# dataframe with important data from each state/union territory as well as India
statewise_data.head()
```

Out[18]:

| | State / Union Territory | Confirmed | Cured | Deaths | Active | Population | Population density (per km2) |
|---|---|---|---|---|---|---|---|
| **0** | Andaman and Nicobar Islands | 33 | 33 | 0 | 0 | 380581 | 46.0 |
| **1** | Andhra Pradesh | 2090 | 1056 | 46 | 988 | 49577103 | 303.0 |
| **2** | Arunachal Pradesh | 1 | 1 | 0 | 0 | 1383727 | 17.0 |
| **3** | Assam | 65 | 39 | 2 | 24 | 31205576 | 397.0 |
| **4** | Bihar | 831 | 383 | 6 | 442 | 104099452 | 1102.0 |

# 3.2 Plot

In [19]:
```
statewise_data.drop(41)[['State / Union Territory', 'Cured', 'Deaths', 'Active']].
plt.title('Total cases in indian states and UTs')
plt.ylabel('Total cases')
plt.tight_layout()
plt.show()
```




# 3.3 Number of cases in india

In [20]:
```
total_cases_national = statewise_data.loc[41]['Confirmed']
active_national = statewise_data.loc[41]['Active']
cured_national = statewise_data.loc[41]['Cured']
deaths_national = statewise_data.loc[41]['Deaths']
print('NUMBER OF CASES IN INDIA\n')
print(f'Total: {total_cases_national}')
print(f'Active: {active_national}')
print(f'Cured" {cured_national}')
print(f'Deaths: {deaths_national}')
```

```
NUMBER OF CASES IN INDIA

Total: 74404
Active: 47600
Cured" 24386
Deaths: 2418
```

# 4 Non - cumulative number of cases (categorised by current health status) grouped by date

4.1 Creating a dataframe with number of cases (categorised by current health status) grouped by date [NON - CUMULATIVE]

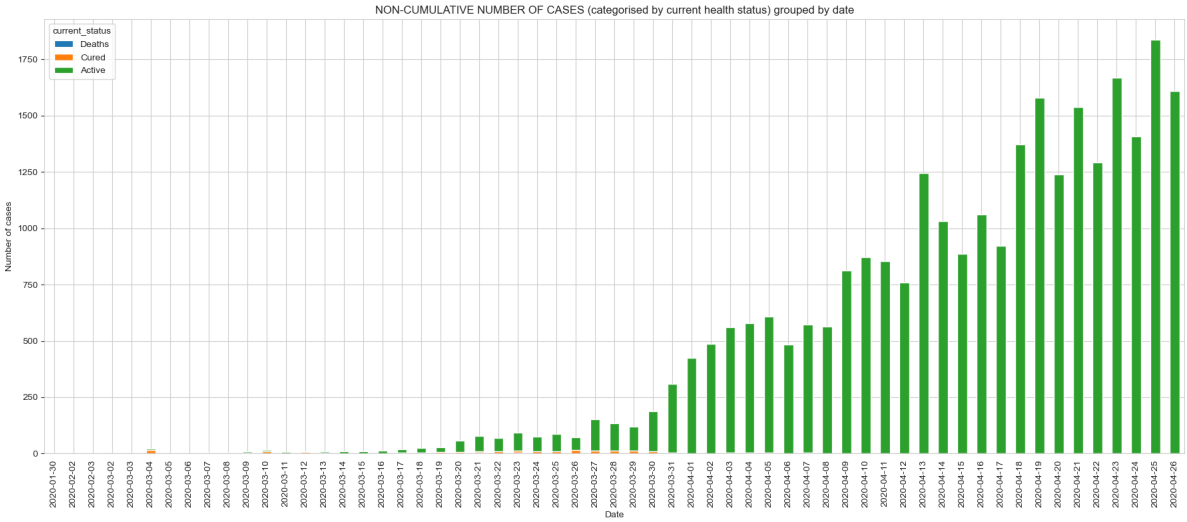
In [21]:
```
datewise_count = individual_details[individual_details['current_status'] != 'Migra
datewise_count = datewise_count.groupby(['diagnosed_date', 'current_status']).coun
datewise_count.fillna(0, inplace = True)
datewise_count.reset_index(inplace = True)

# change the date format to 'YYYY-MM-DD'
datewise_count['diagnosed_date'] = datewise_count['diagnosed_date'].apply(lambda d

datewise_count.sort_values('diagnosed_date', inplace = True)
datewise_count.rename(columns = {'Deceased': 'Deaths', 'Hospitalized': 'Active', '

# calculate the total number of confirmed cases
datewise_count['Confirmed'] = datewise_count['Active'] + datewise_count['Cured'] +

datewise_count['Deaths'] = datewise_count['Deaths'].apply(lambda num : int(num))
datewise_count['Active'] = datewise_count['Active'].apply(lambda num : int(num))
datewise_count['Cured'] = datewise_count['Cured'].apply(lambda num : int(num))
datewise_count['Confirmed'] = datewise_count['Confirmed'].apply(lambda num : int(nu
datewise_count = datewise_count[['diagnosed_date', 'Confirmed', 'Deaths', 'Cured',

# dataframe with number of cases (categorised by current health status) grouped by
datewise_count.head()
```

Out[21]:

| current_status | diagnosed_date | Confirmed | Deaths | Cured | Active |
|---|---|---|---|---|---|
| **56** | 2020-01-30 | 1 | 0 | 1 | 0 |
| **1** | 2020-02-02 | 1 | 0 | 1 | 0 |
| **4** | 2020-02-03 | 1 | 0 | 1 | 0 |
| **2** | 2020-03-02 | 2 | 0 | 2 | 0 |
| **5** | 2020-03-03 | 1 | 0 | 1 | 0 |

# 4.2 plot

In [22]:
```
datewise_count.drop(columns = ['Confirmed']).plot.bar(x= 'diagnosed_date', stacked
plt.title('NON-CUMULATIVE NUMBER OF CASES (categorised by current health status) gr
plt.xlabel('Date')
plt.ylabel('Number of cases')
plt.tight_layout()
plt.show()
```




### 5 Important COVID-19 statistics of india and each of its states and union territories

5.1 calculation from availbale data

In [23]:
```
# calculate total cases per million people for each state/ union Territory and ind
statewise_data['Total cases per million people'] = statewise_data['Confirmed'] / s

# calculation recovery rate (%) for each state / Union Territory and india
statewise_data['Recovery rate(%)'] = statewise_data['Cured'] / statewise_data['Con

# calculate deaths per million people for each state / Union territory and india
statewise_data['Deaths per million people'] = statewise_data['Deaths'] / statewise_

# calculate fatality rate(%) for each state/Union Territory and india
statewise_data['Fatality rate(%)'] = statewise_data['Deaths'] / statewise_data['Co

statewise_data.head()
```

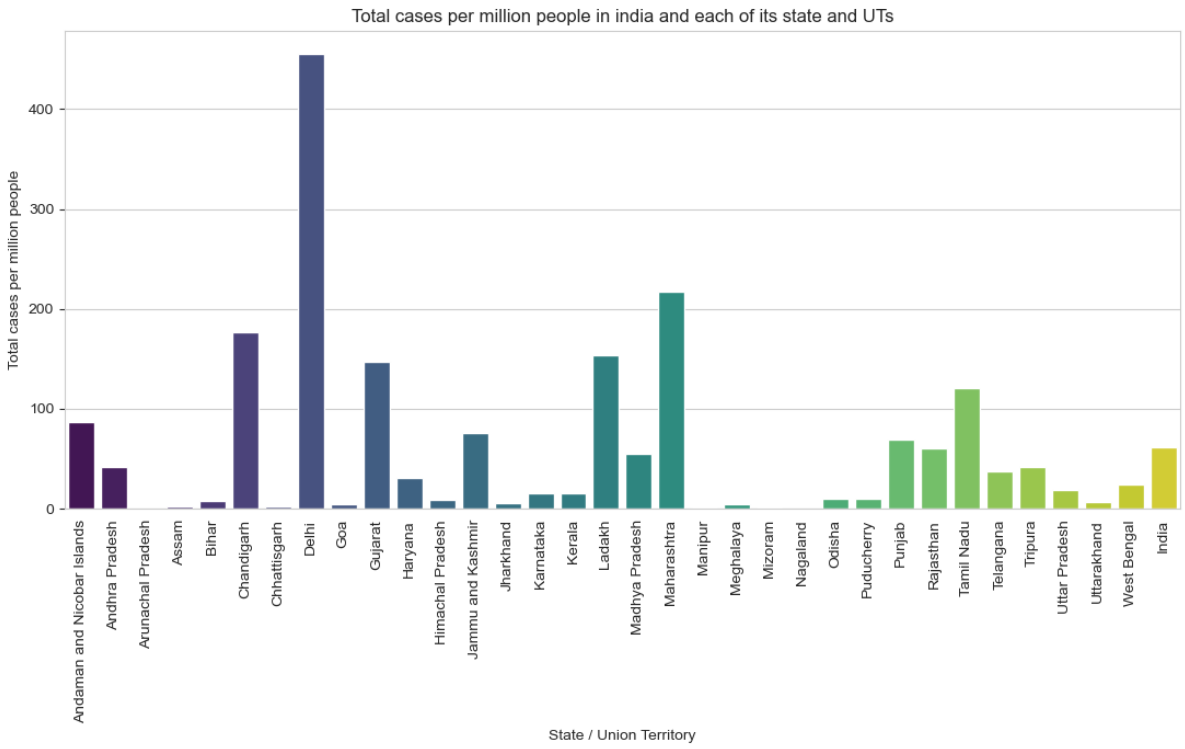
Out[23]:

| | State / Union Territory | Confirmed | Cured | Deaths | Active | Population | Population density (per km2) | Total cases per million people | Recovery rate(%) | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Andaman and Nicobar Islands | 33 | 33 | 0 | 0 | 380581 | 46.0 | 86.709531 | 100.000000 | 0 |
| **1** | Andhra Pradesh | 2090 | 1056 | 46 | 988 | 49577103 | 303.0 | 42.156558 | 50.526316 | 0 |
| **2** | Arunachal Pradesh | 1 | 1 | 0 | 0 | 1383727 | 17.0 | 0.722686 | 100.000000 | 0 |
| **3** | Assam | 65 | 39 | 2 | 24 | 31205576 | 397.0 | 2.082961 | 60.000000 | 0 |
| **4** | Bihar | 831 | 383 | 6 | 442 | 104099452 | 1102.0 | 7.982751 | 46.089049 | 0 |

# 5.2 Total cases per million people(plot)

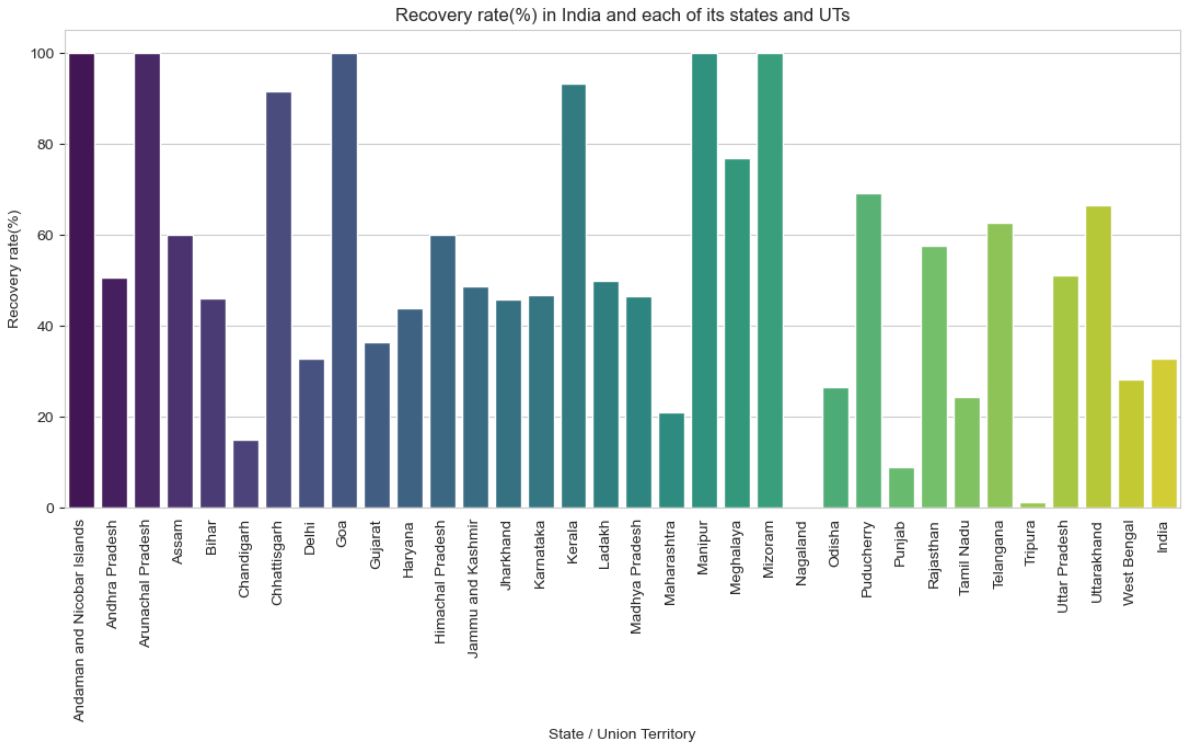
In [24]:
```
plt.figure(figsize=(11,7),dpi =100)
sns.barplot(x = 'State / Union Territory', y = 'Total cases per million people', da
plt.title('Total cases per million people in india and each of its state and UTs')
```

```
plt.xticks(rotation =90)
plt.tight_layout()
plt.show()
```



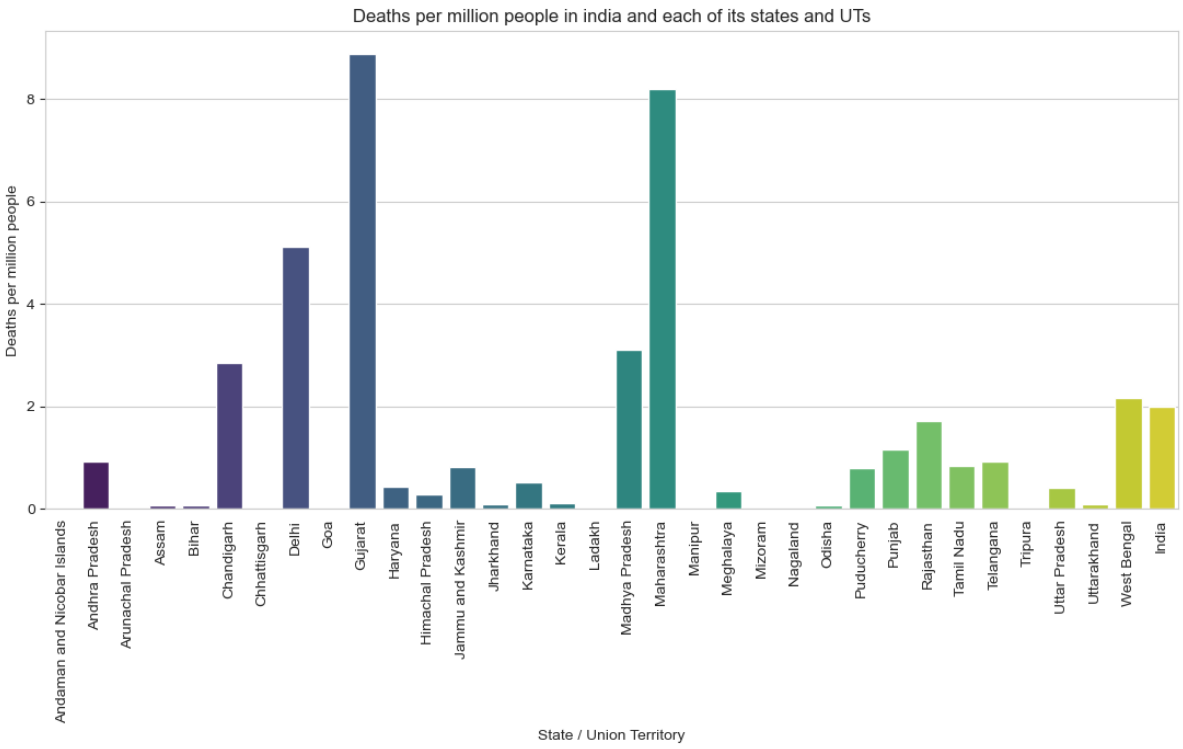

# 5.3 Recovery rate(%) (plot)

In [25]:
```
plt.figure(figsize=(11, 7), dpi = 100)
sns.barplot(x = 'State / Union Territory', y = 'Recovery rate(%)', data = statewis
plt.title('Recovery rate(%) in India and each of its states and UTs')
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```



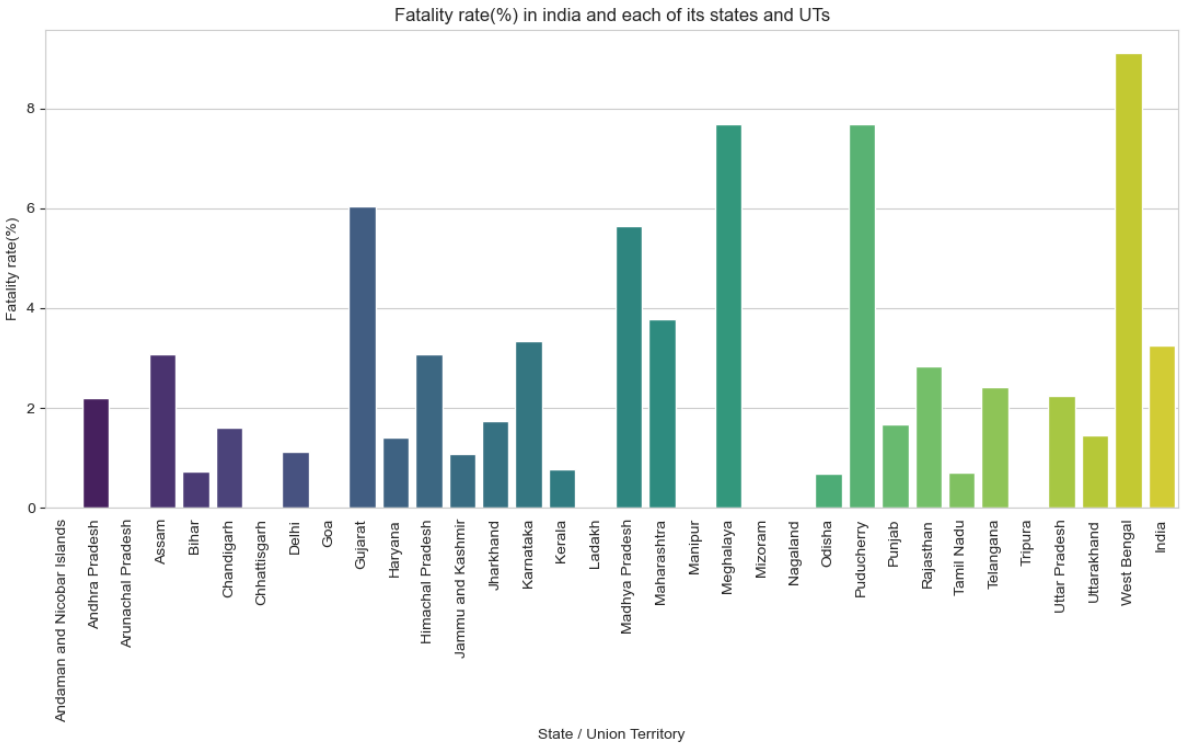
# 5.4 Deaths per million people (plot)

In [26]:
```
plt.figure(figsize=(11,7), dpi = 100)
sns.barplot(x= 'State / Union Territory', y = 'Deaths per million people', data = 
plt.title('Deaths per million people in india and each of its states and UTs')
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```




# 5.5 Fatality rate(%)(plot)

In [27]:
```
plt.figure(figsize=(11,7), dpi = 100)
sns.barplot(x = 'State / Union Territory', y = 'Fatality rate(%)', data = statewis
plt.title('Fatality rate(%) in india and each of its states and UTs')
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```




# 6 Age distribution of positive cases in india and each of its states and union territories.

## 6.1 Data cleaning

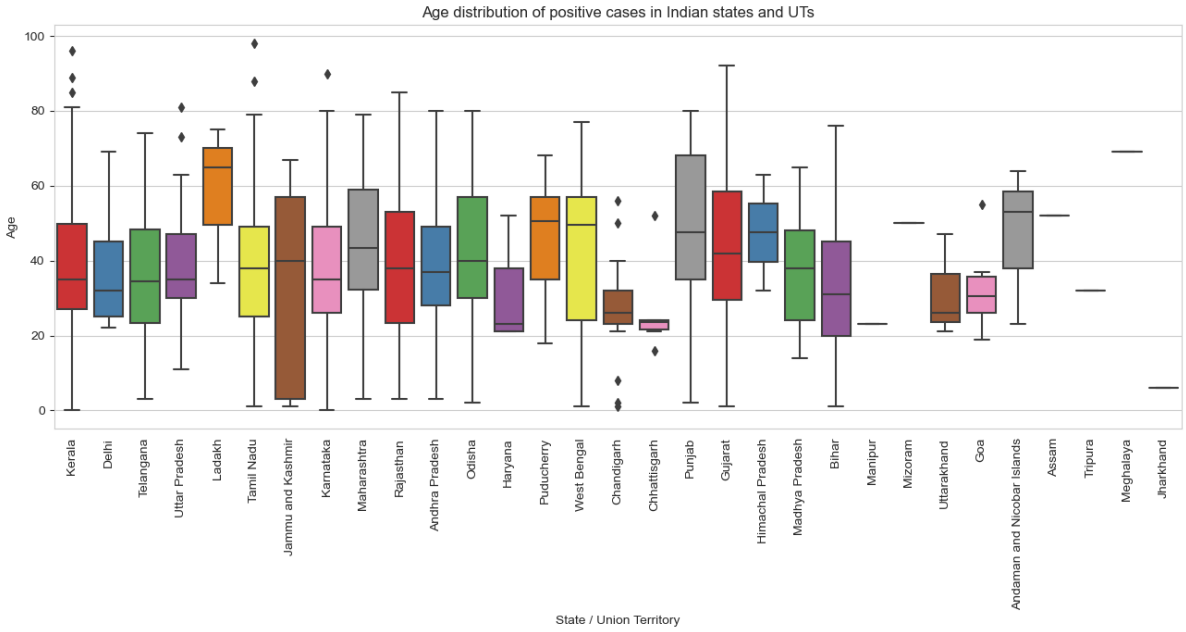
In [28]:
```
# create a list of the Indian states where positive covid-19 cases have been report

list_of_states = list(individual_details['detected_state'].unique())

# create a list of nationalities of all the positive COVID=19 cases reported in ind
list_of_nations = list(individual_details[individual_details['nationality'].isnull

temp_age = individual_details[['age','detected_state', 'nationality', 'current_sta
for i in temp_age.index:
    if temp_age.loc[i]['detected_state'] in list_of_states and temp_age.loc[i]['na
        # assign nationality as 'India', if detected state is in the list of indian
        temp_age.loc[i]['nationality'] = 'India'

# select only those cases whose nationality is mentioned 'India'
temp_age = temp_age[temp_age['nationality']=='India']

# drop the rows where age is mentioned as 'F' or 'M'
temp_age = temp_age[(temp_age['age'] != 'F') & (temp_age['age'] != 'M')]

# drop the rows where ahe is not mentioned
temp_age.dropna(subset = ['age'], inplace = True)


# if age is mentioned as '28-35', update it with the mean value , i.e 31(actual mea
# typecast age given as a string into a floating point number first , and them into
temp_age['age'] = temp_age['age'].apply(lambda age : 3 if age == '28-35' else int(

temp_age.head()
```

Out[28]:

| | age | detected_state | nationality | current_status |
|---|---|---|---|---|
| **0** | 20 | Kerala | India | Recovered |
| **3** | 45 | Delhi | India | Recovered |
| **4** | 24 | Telangana | India | Recovered |
| **23** | 45 | Uttar Pradesh | India | Recovered |
| **25** | 16 | Uttar Pradesh | India | Recovered |

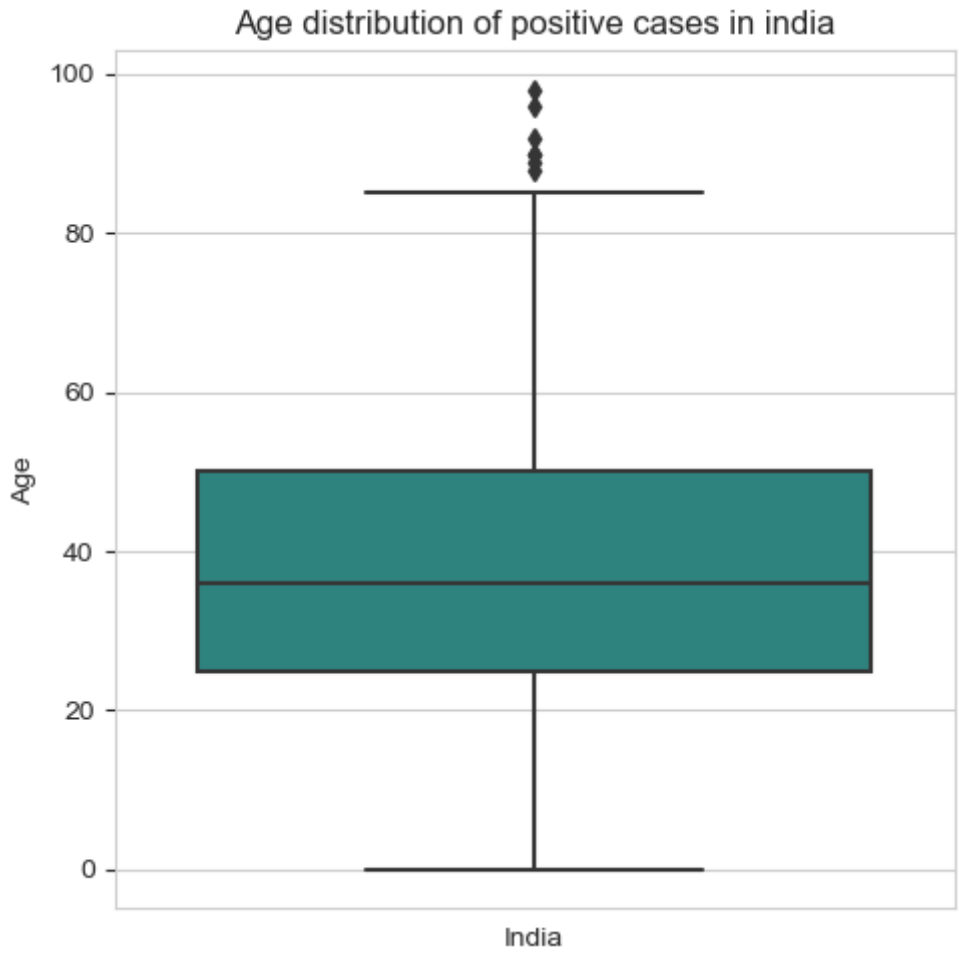
# 6.2 plot (States and Union territories)

In [29]:
```
plt.figure(figsize=(13,7), dpi = 100)
sns.boxplot(x = 'detected_state', y = 'age', data = temp_age, palette = 'Set1')
plt.title('Age distribution of positive cases in Indian states and UTs')
plt.xlabel('State / Union Territory')
plt.ylabel('Age')
plt.xticks(rotation =90)
plt.tight_layout()
plt.show()
```




# 6.3 Plot (india)

In [30]:
```
plt.figure(figsize=(5,5), dpi = 100)
sns.boxplot(x = 'nationality',y = 'age', data =temp_age, palette = 'viridis')
plt.title('Age distribution of positive cases in india')
plt.xlabel('')
plt.ylabel('Age')
plt.tight_layout()
plt.show()
```




# 7 National descriptive statistics (Age)

# 7.1 Ages of all positive cases in india

In [31]:
```
print('NATIONAL DESCRIPTIVE STATISTICS (Age of all positive cases in india)\n')
print(temp_age.describe()['age'])
```

```
NATIONAL DESCRIPTIVE STATISTICS (Age of all positive cases in india)

count    2318.000000
mean       38.234254
std        17.283662
min         0.000000
25%        25.000000
50%        36.000000
75%        50.000000
max        98.000000
Name: age, dtype: float64
```

# 7.2 Ages of positive cases in india (grouped by current health status)

In [32]:
```
print('NATIONAL DESCRIPTIVE STATISTICS (Age of positive cases in India , grouped by
print(temp_age.groupby('current_status').describe()['age'])
```

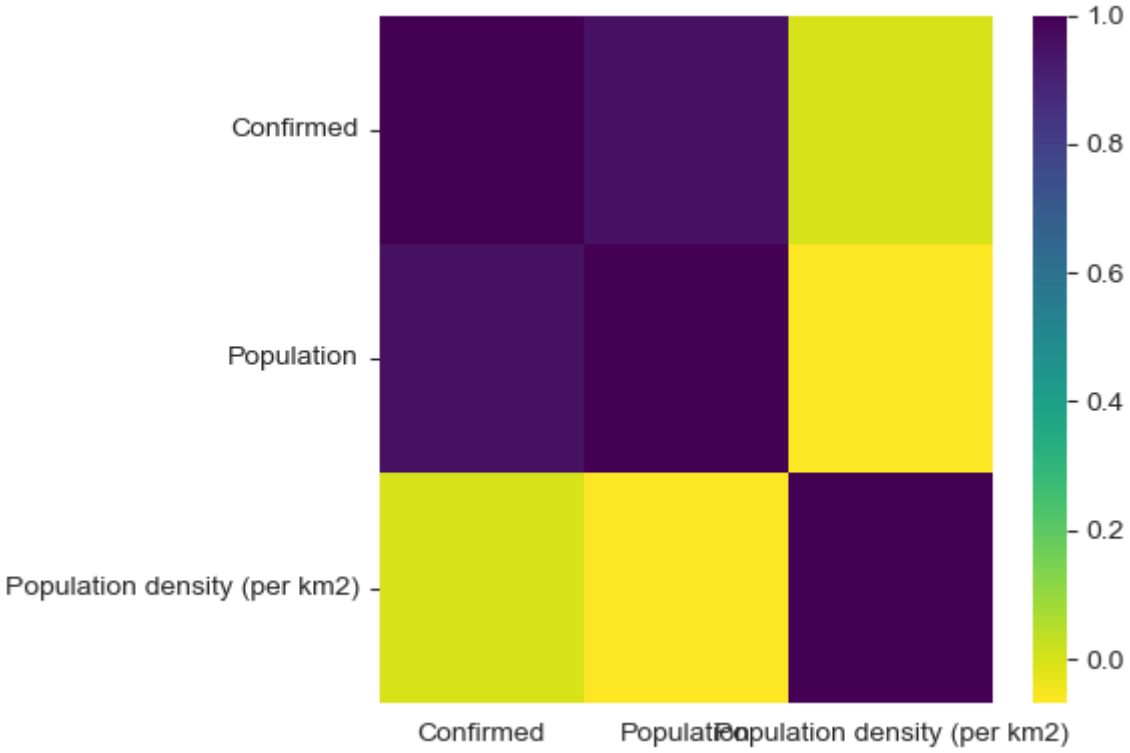
```
NATIONAL DESCRIPTIVE STATISTICS (Age of positive cases in India , grouped by curre
nt health status )
```

| current_status | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| Deceased | 43.0 | 59.860465 | 16.359912 | 1.0 | 50.0 | 65.0 | 70.00 | 85.0 |
| Hospitalized | 2165.0 | 37.783372 | 16.970881 | 0.0 | 25.0 | 36.0 | 50.00 | 98.0 |
| Recovered | 110.0 | 38.654545 | 18.443783 | 1.0 | 24.0 | 35.0 | 50.25 | 96.0 |

# 8 Correlation between population (or, population desity) and total positive cases

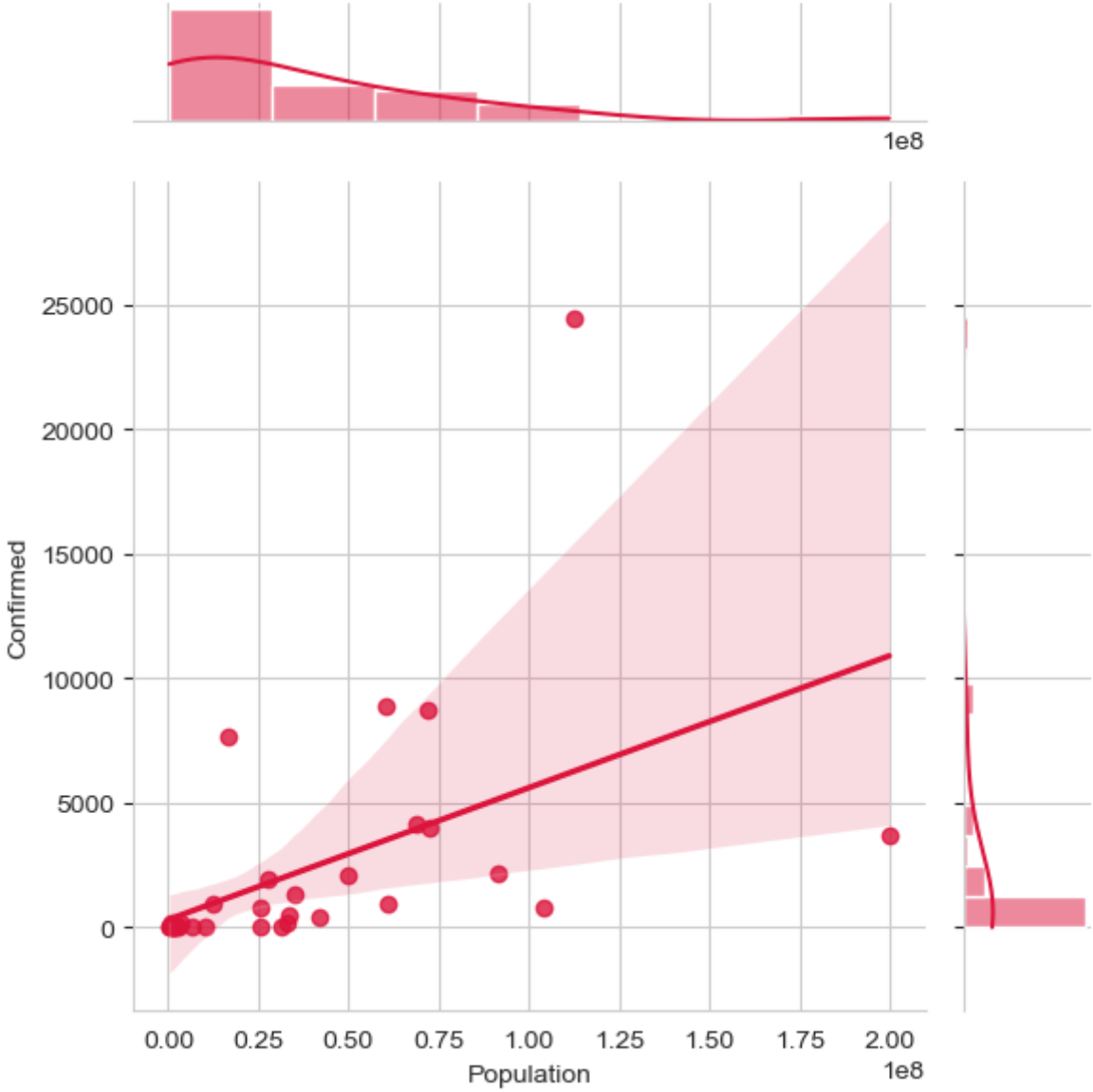
8.1 Heat map

In [33]:
```
plt.figure(figsize=(6,4), dpi = 100)
sns.heatmap(statewise_data[['Confirmed','Population', 'Population density (per km2
plt.yticks(rotation = 0)
plt.tight_layout()
plt.show()
```



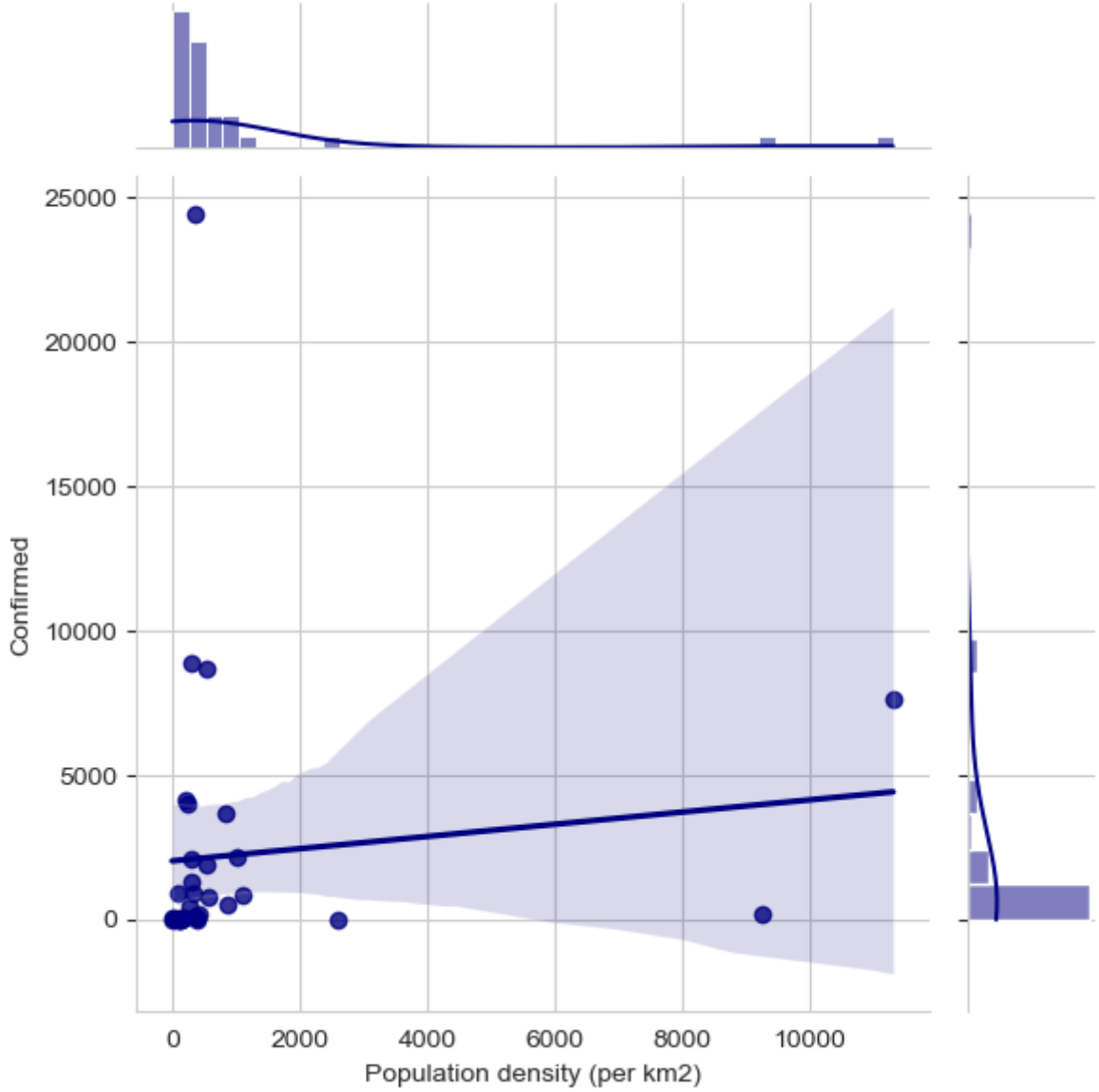

# 8.2 Simple linear regression (independent variable: population , dependent variable: total positive cases)

In [34]:
```
sns.jointplot(x = 'Population', y = 'Confirmed', data = statewise_data.drop(41), k
plt.ylabel('Total positive cases')
plt.tight_layout()
plt.show()
```




# 8.3 simple linear regression (independent variable : Density per km2, dependent variable : Total positive cases)

In [35]:
```
sns.jointplot(x = 'Population density (per km2)', y = 'Confirmed', data = statewis
plt.ylabel('Total positive cases')
plt.tight_layout()
plt.show()
```




# 9 public health facilities in india and each of its states and union territories

## 9.1 Creating a dataframe with public health facility details for each state / UT and india.

In [36]:
```
hospital_details = hospital_beds_india[['State/UT', 'TotalPublicHealthFacilities_HM
hospital_details.rename(columns = {'State/UT': 'State / Union Territory'}, inplace
hospital_details['State / Union Territory'] = hospital_details['State / Union Terri
hospital_details.dropna(inplace = True)
hospital_details['TotalPublicHealthFacilities_HMIS'] = hospital_details['TotalPubli
hospital_details['NumPublicBeds_HMIS'] = hospital_details['NumPublicBeds_HMIS'].app

# merge (or, add) the details for Dadra and Nagar Haveli & Daman and Diu (as the fo
dnhdd = hospital_details[(hospital_details['State / Union Territory'] == 'Dadra and
hospital_details.drop(index = [7, 8, 36], inplace = True)
hospital_details.loc[36] = ['Dadra and Nagar Haveli and Daman and Diu', dnhdd['Tota

hospital_details.sort_values('State / Union Territory', inplace = True)
hospital_details.loc[37] = ['India', hospital_details['TotalPublicHealthFacilities_
hospital_details = pd.merge(left = hospital_details, right = temp_population.drop(c
```

```
# calculate public health facility details per 1000 people for each state/UT as wel
hospital_details['TotalPublicHealthFacilities/1000 people'] = (hospital_details['To
hospital_details['NumPublicBeds/1000 people'] = (hospital_details['NumPublicBeds_HM

# dataframe with public health facility details for each state/UT and India
hospital_details.head()
```

```
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\2037430088.py:2: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  hospital_details.rename(columns = {'State/UT': 'State / Union Territory'}, inpla
ce = True)
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\2037430088.py:3: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  hospital_details['State / Union Territory'] = hospital_details['State / Union Te
rritory'].apply(lambda name : str(name).replace('&', 'and'))
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\2037430088.py:4: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  hospital_details.dropna(inplace = True)
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\2037430088.py:5: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  hospital_details['TotalPublicHealthFacilities_HMIS'] = hospital_details['TotalPu
blicHealthFacilities_HMIS'].apply(lambda count : int(str(count).replace(',', '')))
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\2037430088.py:6: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  hospital_details['NumPublicBeds_HMIS'] = hospital_details['NumPublicBeds_HMIS'].
apply(lambda count : int(str(count).replace(',', '')))
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\2037430088.py:10: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  hospital_details.drop(index = [7, 8, 36], inplace = True)
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\2037430088.py:11: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  hospital_details.loc[36] = ['Dadra and Nagar Haveli and Daman and Diu', dnhdd['T
otalPublicHealthFacilities_HMIS'], dnhdd['NumPublicBeds_HMIS']]
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\2037430088.py:13: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  hospital_details.sort_values('State / Union Territory', inplace = True)
```

```
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\2037430088.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  hospital_details.loc[37] = ['India', hospital_details['TotalPublicHealthFacilities_HMIS'].sum(), hospital_details['NumPublicBeds_HMIS'].sum()]
```

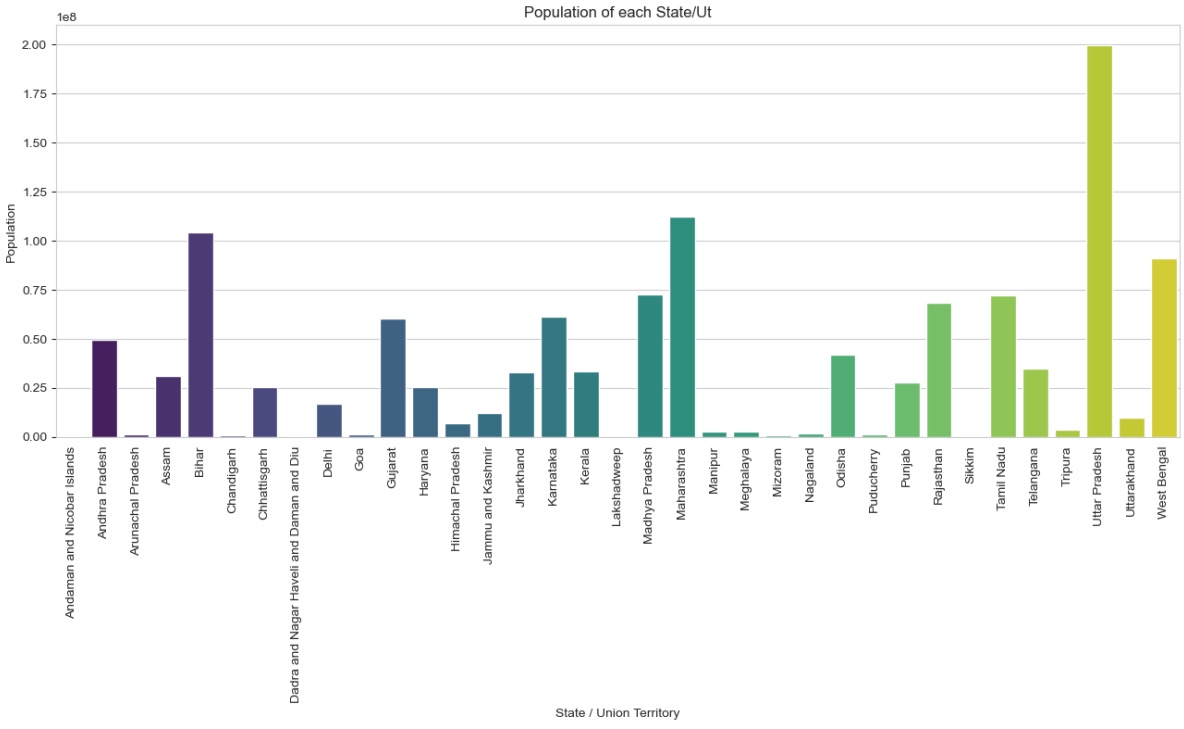
Out[36]:

| | State / Union Territory | TotalPublicHealthFacilities_HMIS | NumPublicBeds_HMIS | Population | TotalPublicHealth |
|---|---|---|---|---|---|
| 0 | Andaman and Nicobar Islands | 34 | 1246 | 380581 | |
| 1 | Andhra Pradesh | 1666 | 60799 | 49577103 | |
| 2 | Arunachal Pradesh | 199 | 2320 | 1383727 | |
| 3 | Assam | 1220 | 19115 | 31205576 | |
| 4 | Bihar | 2146 | 17796 | 104099452 | |

# 9.2 Populatiojn of each state/UT (plot)

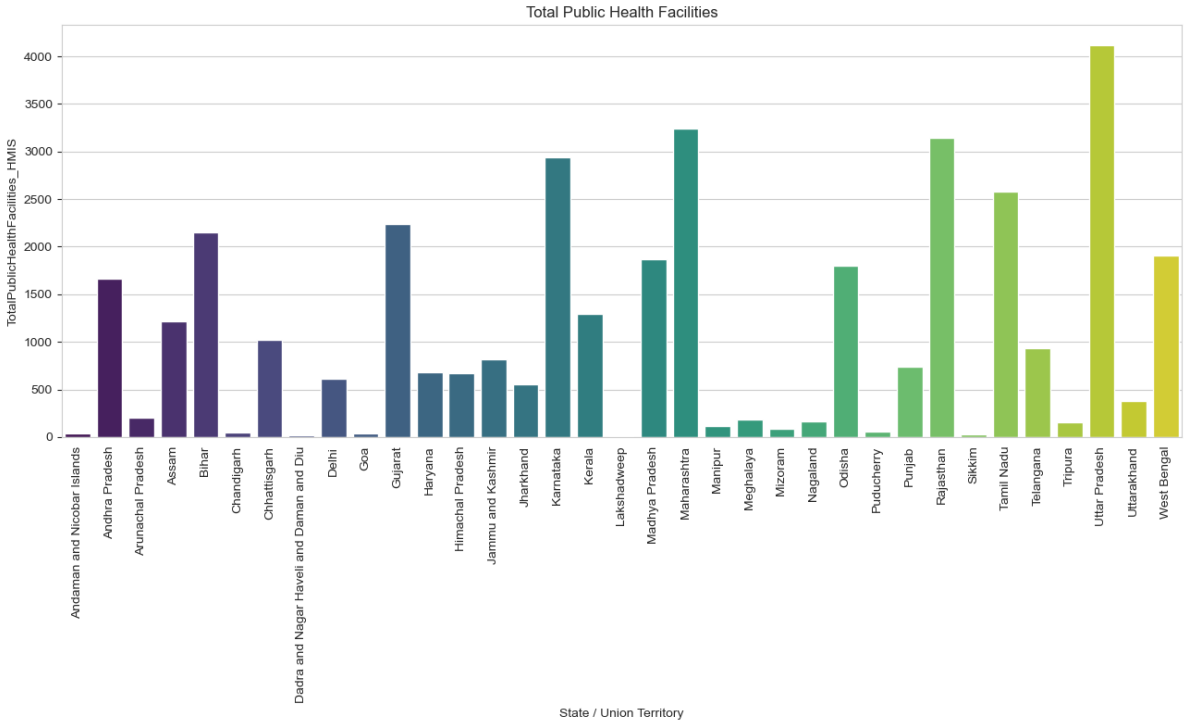
In [37]:

```
plt.figure(figsize=(13,8), dpi = 100)
sns.barplot(x = 'State / Union Territory', y = 'Population', data = hospital_detai
plt.title('Population of each State/Ut')
plt.xticks(rotation =90)
plt.tight_layout()
plt.show()
```



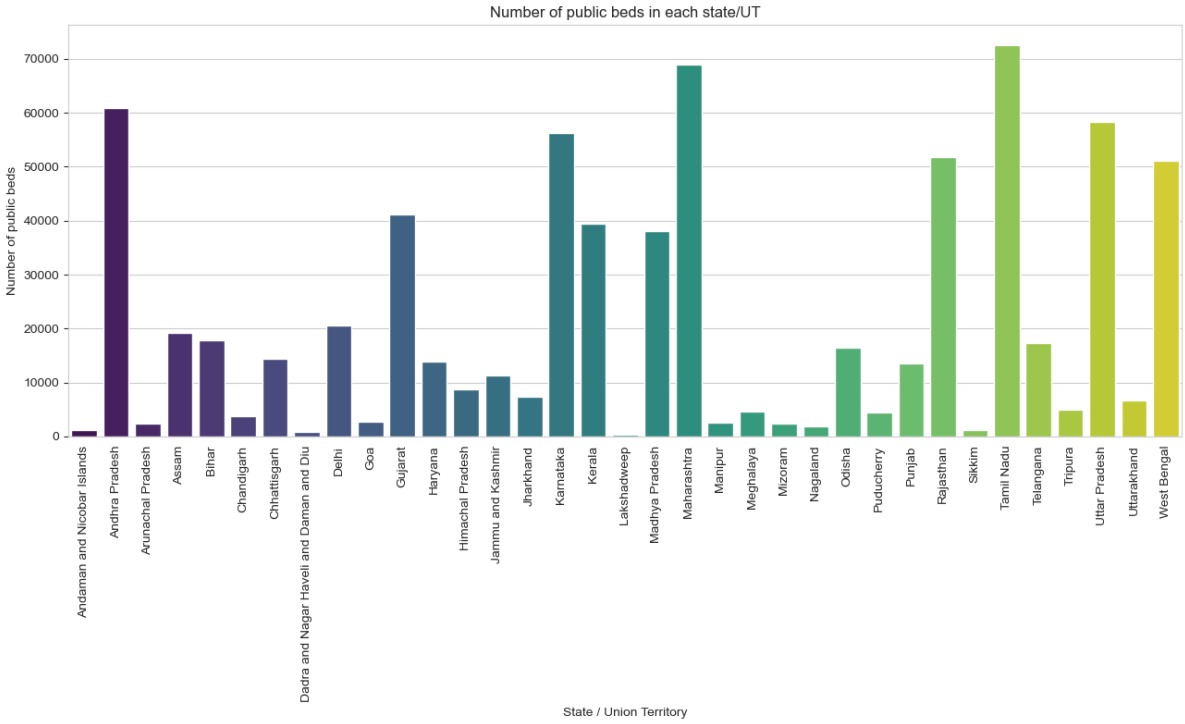
# 9.3 Total public health facilities in each state/ UT (plot)

In [38]:
```
plt.figure(figsize=(13,8), dpi = 100)
sns.barplot(x = 'State / Union Territory', y = 'TotalPublicHealthFacilities_HMIS',
plt.title('Total Public Health Facilities')
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```



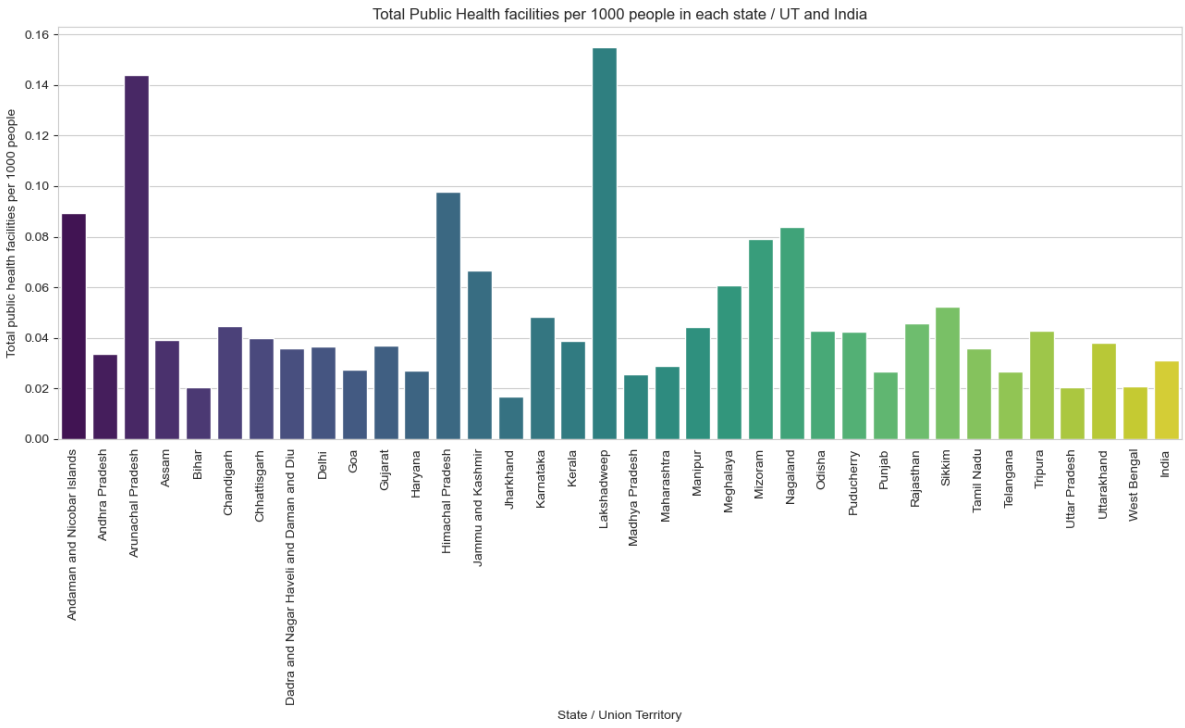

# 9.4 Number of public beds in each state / UT (plot)

In [39]:
```
plt.figure(figsize=(13, 8), dpi = 100)
sns.barplot(x = 'State / Union Territory', y = 'NumPublicBeds_HMIS', data = hospita
plt.title('Number of public beds in each state/UT')
plt.ylabel('Number of public beds')
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```



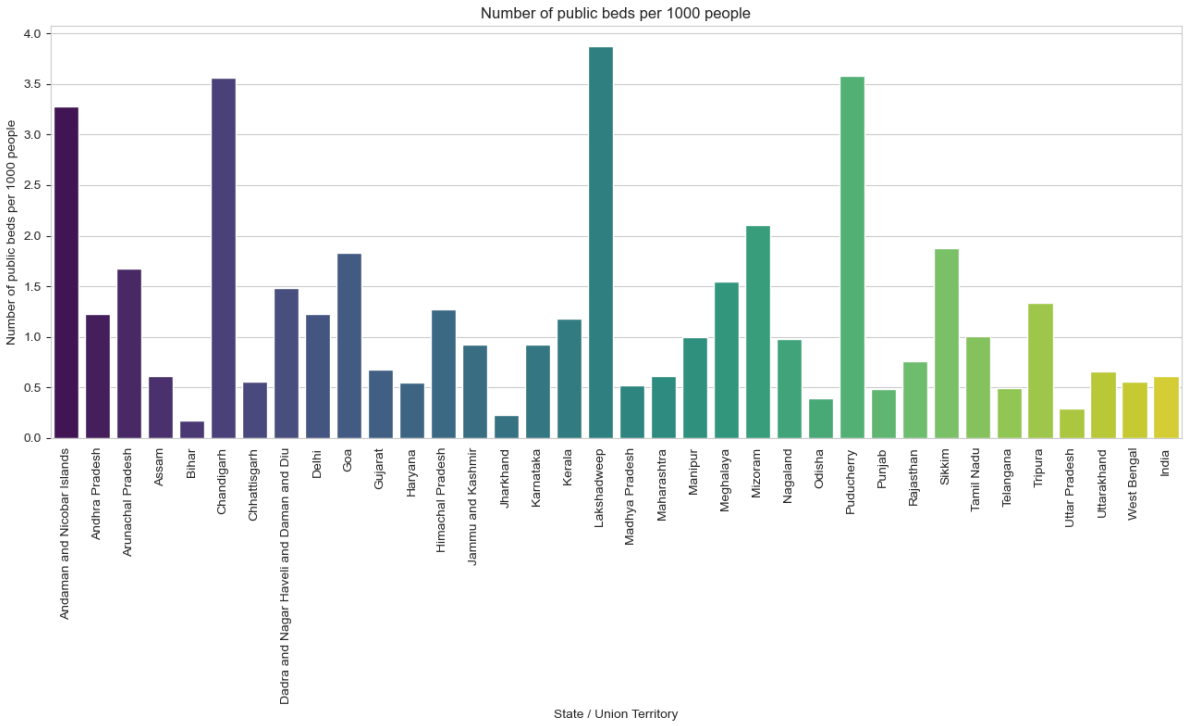

# 9.5 Total public health facilities per 1000 people in each state / UT and india (plot)

In [40]:
```
plt.figure(figsize=(13,8), dpi = 100),
sns.barplot(x= 'State / Union Territory', y = 'TotalPublicHealthFacilities/1000 pe
plt.title('Total Public Health facilities per 1000 people in each state / UT and I
plt.ylabel('Total public health facilities per 1000 people')
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```



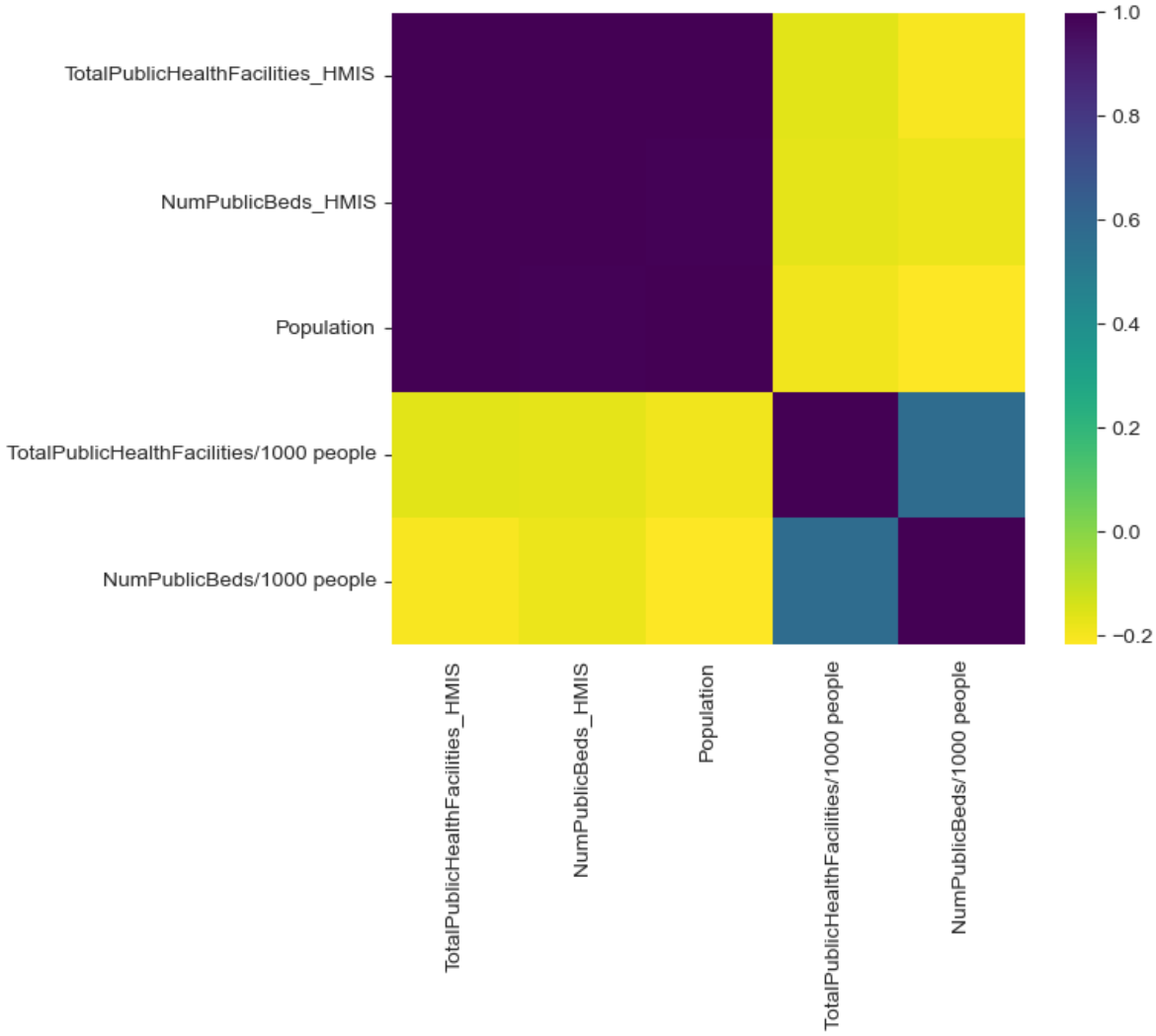
# 9.6 Number of public beds per 1000 people in each state / UT and india (plot)

In [41]:
```
plt.figure(figsize=(13, 8), dpi = 100)
sns.barplot(x = 'State / Union Territory', y = 'NumPublicBeds/1000 people', data =
plt.title('Number of public beds per 1000 people')
plt.ylabel('Number of public beds per 1000 people')
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```




# 9.7 Correlation heat map

In [42]:
```
plt.figure(figsize=(8,7), dpi = 100)
sns.heatmap(hospital_details.corr(), cmap = 'viridis_r')
plt.tight_layout()
plt.show()
```




# 10 Indian Council Of Medical Research (ICMR) testing details:

##10.1 Data Cleaning

In [43]:
```
testing_details = icmr_testing_details[['TotalSamplesTested', 'TotalPositiveCases'

# change the date format to 'YYYY-MM-DD'
testing_details['Date'] = icmr_testing_details['DateTime'].apply(lambda  dt : '20'+

testing_details = testing_details[['Date', 'TotalSamplesTested', 'TotalPositiveCas

# dataframe with date-wise testing details provided by the Indian Council Of Medic
testing_details.head()
```

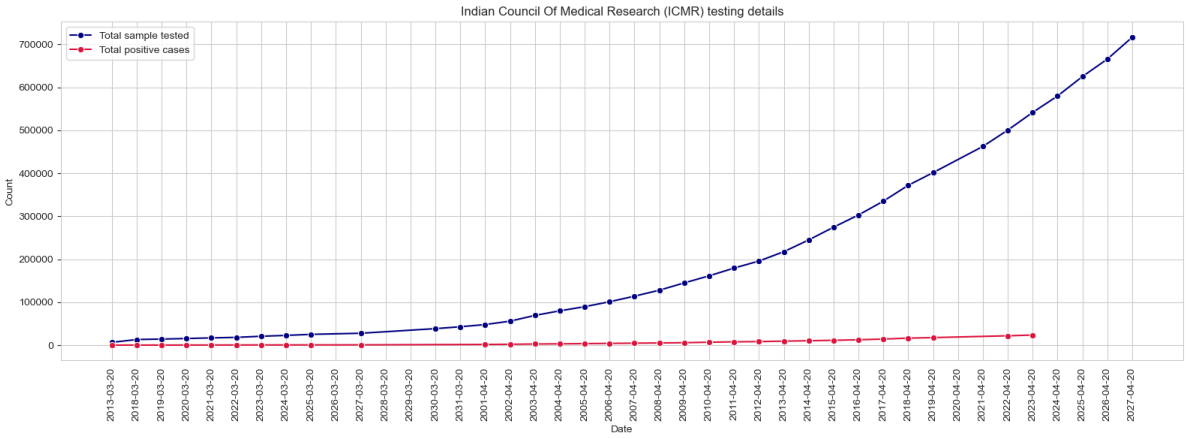
```
C:\Users\meanu\AppData\Local\Temp\ipykernel_18472\1500444293.py:4: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  testing_details['Date'] = icmr_testing_details['DateTime'].apply(lambda  dt : '2
0'+ '-'.join(dt.split(' ')[0].split('/'[::-1])))
```

Out[43]:

| | Date | TotalSamplesTested | TotalPositiveCases |
|---|---|---|---|
| **0** | 2013-03-20 | 6500.0 | 78.0 |
| **1** | 2018-03-20 | 13125.0 | 150.0 |
| **2** | 2019-03-20 | 14175.0 | 182.0 |
| **3** | 2020-03-20 | 15404.0 | 236.0 |
| **4** | 2021-03-20 | 16911.0 | 315.0 |

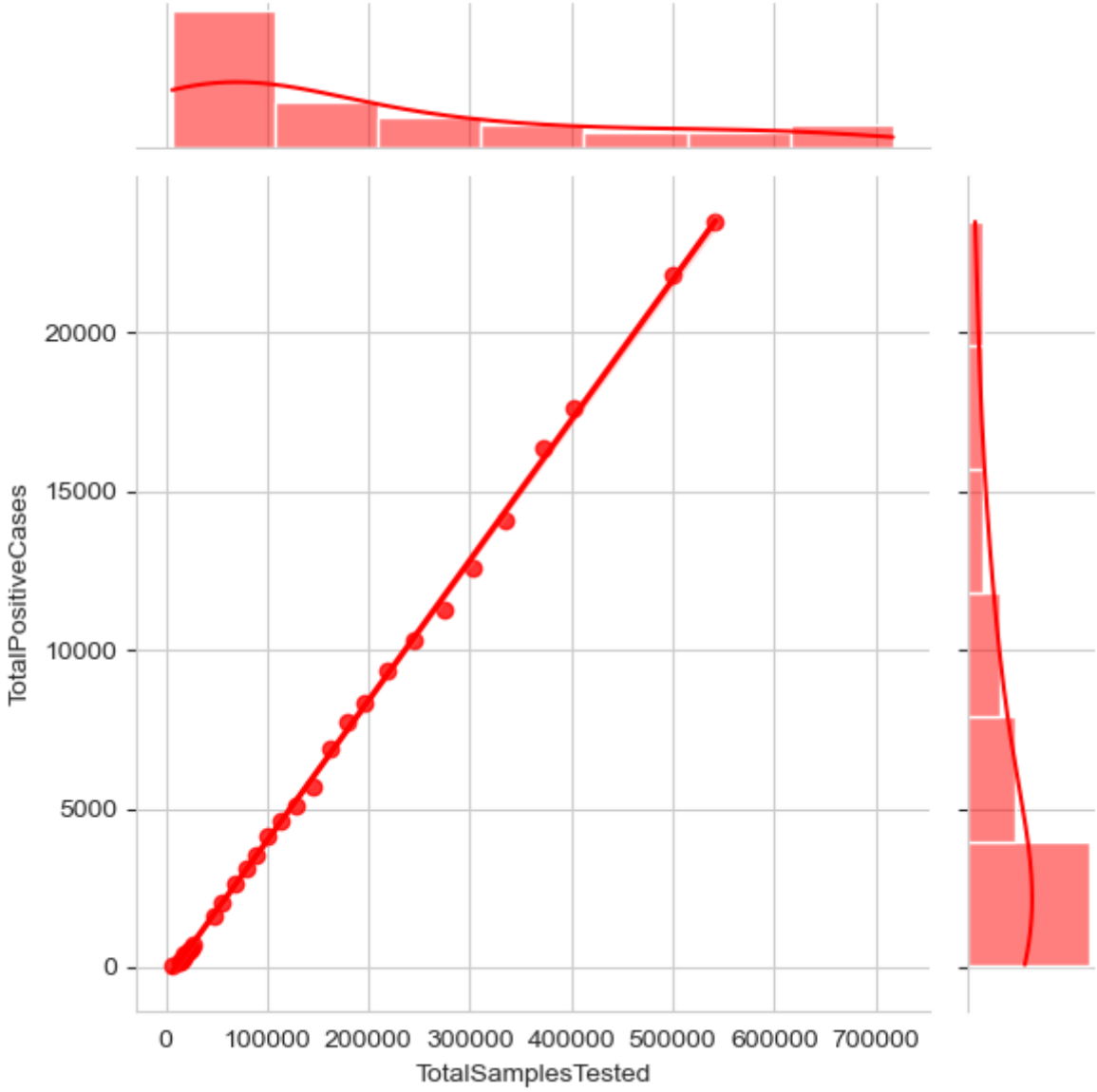
# 10.2 Total sample tested and total positiv cases reported on each date (plot)

In [44]:
```
plt.figure(figsize=(16,6), dpi = 100)
sns.lineplot(x = 'Date', y = 'TotalSamplesTested', data = testing_details,label =
sns.lineplot(x ='Date', y = 'TotalPositiveCases', data = testing_details, label =
plt.title('Indian Council Of Medical Research (ICMR) testing details')
plt.ylabel('Count')
plt.legend(loc = 0)
plt.xticks(rotation =90)
plt.tight_layout()
plt.show()
```




# 10.3 Simple linear regression (independent varible : Total sample tested , dependent variable: Total positive cases)

In [45]:
```
sns.jointplot(x = 'TotalSamplesTested', y = 'TotalPositiveCases', data = testing_d
plt.xlabel('Total samples tested')
plt.ylabel('Total positive cases')
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```




# 11 State/Union territories and districts with the highest and lowest number of confirmed COVID-19 cases

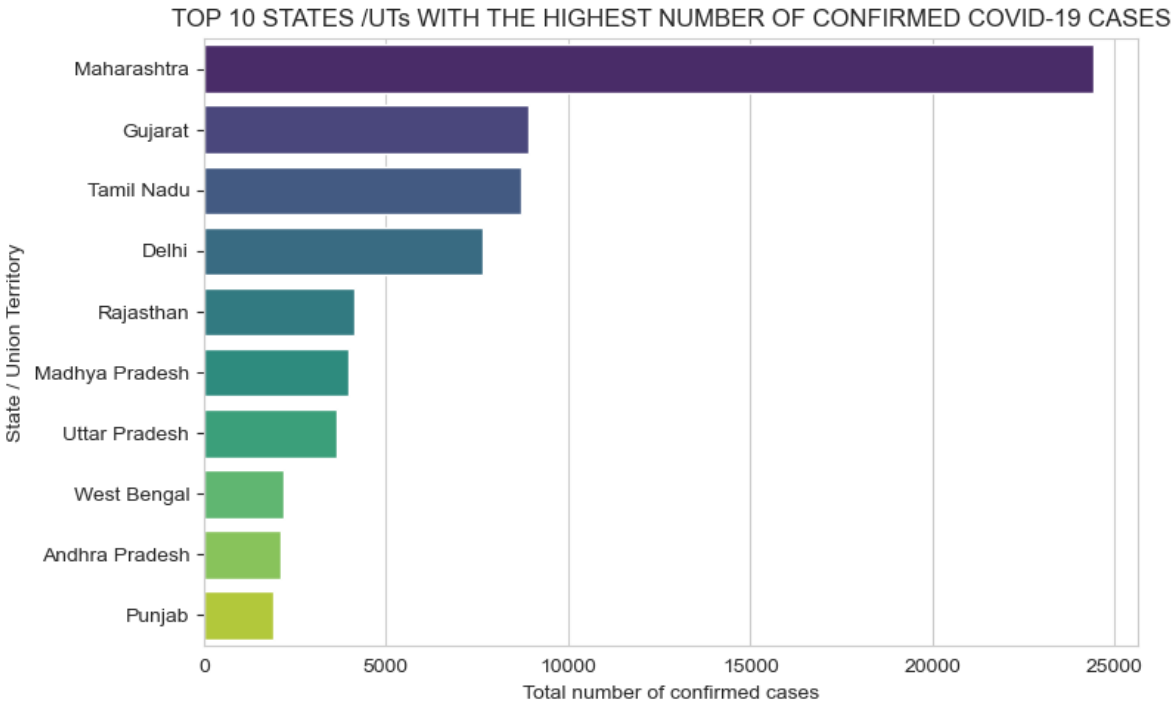
11.1 Data cleaning and creation of appropriate series

In [46]:
```
# series : states /UTs (highest number of confirmed COVID-19 cases)
states_highest = statewise_data.groupby('State / Union Territory').mean()['Confirm

# state: state / UTs (lowest number of confirmed COVID-19 cases)
states_lowest = statewise_data.groupby('State / Union Territory').mean()['Confirme

# series districts (highest number of confirmed COVID -19 cases)
district_highest= individual_details['detected_district'].value_counts().head(10)
```

# 11.2 States/ UTs (highest number of confirmed COCID-19 cases)

(A) plot

In [47]:
```
plt.figure(figsize=(8, 5), dpi = 100)
sns.barplot(x = states_highest.values, y = states_highest.index, palette = 'viridis
plt.title('TOP 10 STATES /UTs WITH THE HIGHEST NUMBER OF CONFIRMED COVID-19 CASES'
plt.xlabel('Total number of confirmed cases')
plt.ylabel('State / Union Territory')
plt.tight_layout()
plt.show()
```




# (B) Data table

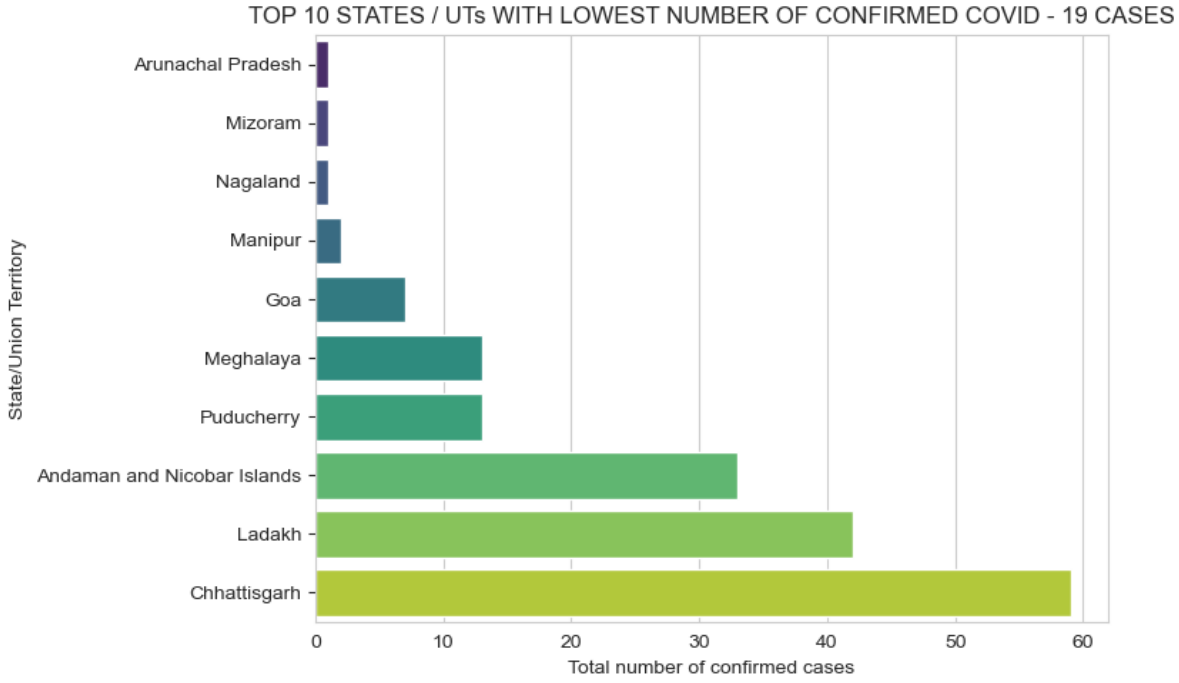
In [48]:
```
print('TOP 10 STATES/UTs WITH THE HIGHEST NUMBER OF CONFIRMED COVID-19 CASES\n')
print(states_highest)
```

```
TOP 10 STATES/UTs WITH THE HIGHEST NUMBER OF CONFIRMED COVID-19 CASES

State / Union Territory
Maharashtra       24427.0
Gujarat            8903.0
Tamil Nadu         8718.0
Delhi              7639.0
Rajasthan          4126.0
Madhya Pradesh     3986.0
Uttar Pradesh      3664.0
West Bengal        2173.0
Andhra Pradesh     2090.0
Punjab             1914.0
Name: Confirmed, dtype: float64
```

# 11.3 States / UTs (lowest number of confirmed COVID-19 cases)

(A)plot

In [49]:
```
plt.figure(figsize=(8, 5), dpi = 100)
sns.barplot(x = states_lowest.values, y = states_lowest.index , palette = 'viridis
plt.title('TOP 10 STATES / UTs WITH LOWEST NUMBER OF CONFIRMED COVID - 19 CASES')
plt.xlabel('Total number of confirmed cases')
plt.ylabel('State/Union Territory')
plt.tight_layout()
plt.show()
```




# (B) Data table

In [50]:
```
print('TOP 10 STATES / UTs WITH THE LOWEST NUMBER OF CONFIRMED COVID-19 CASES\n')
print(states_lowest)
```

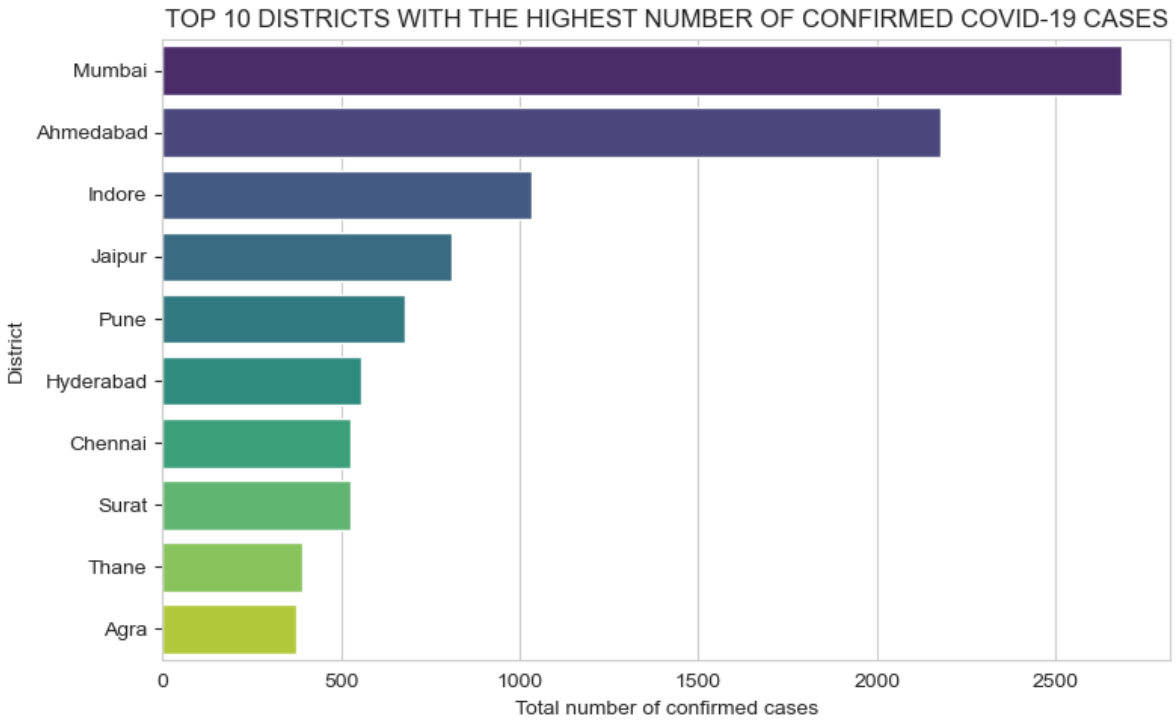
```
TOP 10 STATES / UTs WITH THE LOWEST NUMBER OF CONFIRMED COVID-19 CASES

State / Union Territory
Arunachal Pradesh               1.0
Mizoram                         1.0
Nagaland                        1.0
Manipur                         2.0
Goa                             7.0
Meghalaya                      13.0
Puducherry                     13.0
Andaman and Nicobar Islands    33.0
Ladakh                         42.0
Chhattisgarh                   59.0
Name: Confirmed, dtype: float64
```

# 11.4 Districts (highest number of conformed COVID-19 cases)

(A) plot

In [51]:
```
plt.figure(figsize=(8, 5), dpi = 100)
sns.barplot(x = district_highest.values, y = district_highest.index, palette = 'vi
```

```
plt.title('TOP 10 DISTRICTS WITH THE HIGHEST NUMBER OF CONFIRMED COVID-19 CASES')
plt.xlabel('Total number of confirmed cases')
plt.ylabel('District')
plt.tight_layout()
plt.show()
```




# (B) Data table

In [52]:
```
print('TOP 10 DISTRICTS WITH THE HIGHEST NUMBER OF CONFIRMED COVID -19 CASES\n')
print(district_highest)
```

```
TOP 10 DISTRICTS WITH THE HIGHEST NUMBER OF CONFIRMED COVID -19 CASES

Mumbai       2687
Ahmedabad    2181
Indore       1036
Jaipur        808
Pune          680
Hyderabad     557
Chennai       528
Surat         526
Thane         392
Agra          374
Name: detected_district, dtype: int64
```

# 11.5 District (Lowest number if confirmed COVID-19 cases)

In [53]:
```
print('DISTRICTS WITH ONLY 1 CONFIRMED COVID-19 CASE\n')
for dist, cnt in zip(individual_details['detected_district'].value_counts().index,
    # check if there's only 1 positive case in the district and the district name d
    if cnt == 1 and '*' not in dist:
        print(dist)
```

DISTRICTS WITH ONLY 1 CONFIRMED COVID-19 CASE

Kalimpong
Kathua
Hooghly
Dohad
Siddipet
Imphal East
Washim
Puri
Jehanabad
Lakhisarai
Howrah
Kamrup
Lakhimpur
Golaghat
Tapi
South Salmara Mankachar
Chittorgarh
Nanded
South 24 Parganas
Rajsamand
Ayodhya
Madhepura
East Delhi
Shahjahanpur
Balrampur
Surendranagar
Cuttack
Jangoan
Pudukkottai
The Dangs
Dharmapuri
Ramban
Lohit
Badgam
North Tripura
Mahabubabad
Alirajpur
Pauri Garhwal
Gomati
Parbhani
Hailakandi
Giridih
Fatehabad
Aizawl
Imphal West
Sirmaur
Dhenkanal
Sri Muktsar Sahib
Beed
Bhadohi
Koderma
Charkhi Dadri
Rajnandgaon
Mahe
Bilaspur
Mau
Gondia
Kishtwar
North and Middle Andaman
Sindhudurg
Gonda
Ferozepur

```
Barabanki
North East Delhi
Jamnagar
Unnao
Kodagu
Morbi
Durg
Almora
Jalaun
```

# 12 Date - wise growth rate(%) of conformed casesin India

12.1 Calculation from available data

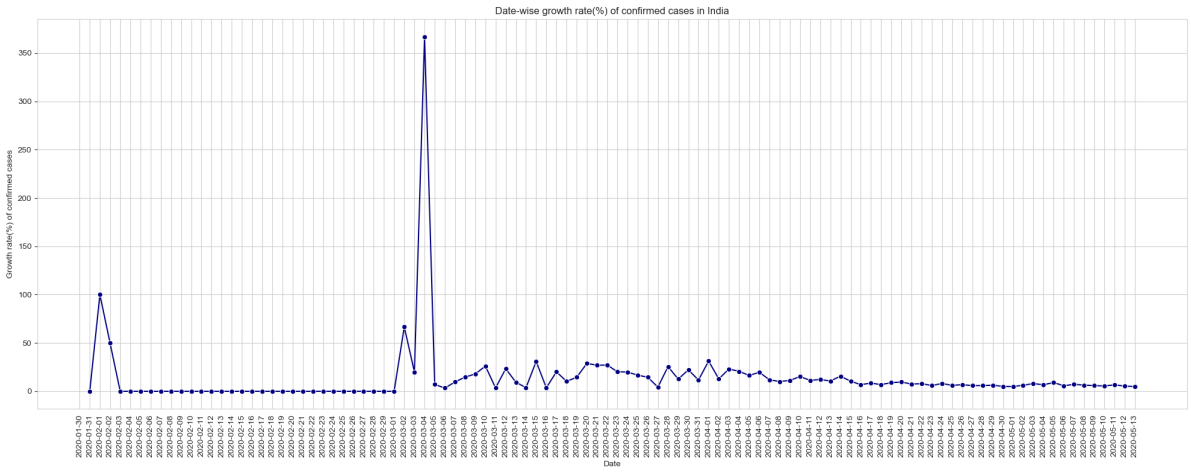
In [54]:
```
date_cumulative.reset_index(inplace = True)
date_cumulative.drop(columns = ['index'], inplace =True)

# Create a list to contain the date-wise growth rate(%) of confirmed cases; the gr
growth_rate = [np.nan]

#Calculate the growth rate for each date (except the first date) given in the data
for i in range (1,len(date_cumulative['Date'])):
    growth_rate.append((date_cumulative.iloc[i]['Confirmed'] - date_cumulative.iloc
# create a new column to store the growth rate for each date given in the data fra
date_cumulative['Growth rate(%)'] = growth_rate

date_cumulative.head()
```

Out[54]:

| | Date | Confirmed | Cured | Deaths | Active | Growth rate(%) |
|---|---|---|---|---|---|---|
| **0** | 2020-01-30 | 1 | 0 | 0 | 1 | NaN |
| **1** | 2020-01-31 | 1 | 0 | 0 | 1 | 0.0 |
| **2** | 2020-02-01 | 2 | 0 | 0 | 2 | 100.0 |
| **3** | 2020-02-02 | 3 | 0 | 0 | 3 | 50.0 |
| **4** | 2020-02-03 | 3 | 0 | 0 | 3 | 0.0 |

# 12.2 Plot

In [55]:
```
plt.figure(figsize=(20, 8), dpi = 100)
sns.lineplot(x = 'Date', y = 'Growth rate(%)', data = date_cumulative, color = 'na
plt.title('Date-wise growth rate(%) of confirmed cases in India')
plt.ylabel('Growth rate(%) of confirmed cases')
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```




# 13.1 Creating a dataframe with testing details of India and each of its states and UTs

In [56]:
```
statewise_testing_details.rename(columns = {'State': 'State / Union Territory'}, i
statewise_testing_details = statewise_testing_details.groupby('State / Union Terri
statewise_testing_details.reset_index(inplace = True)

# add population of each state /UTs to the existing dataframe to create a new merge
statewise_testing_details = pd.merge(left = statewise_testing_details[['State / Un

# CAlculate and add india's testing details to the dataframe
statewise_testing_details.iloc[-1]= ['India', statewise_testing_details['TotalSamp

# calculate total samples per million people for each state / union territory and
statewise_testing_details['Total samples per million people'] = statewise_testing_

# calculate positive cases per 1000 samples for each state/ union territory and In
statewise_testing_details['Positive cases per 1000 samples'] = statewise_testing_d

statewise_testing_details.head()
```

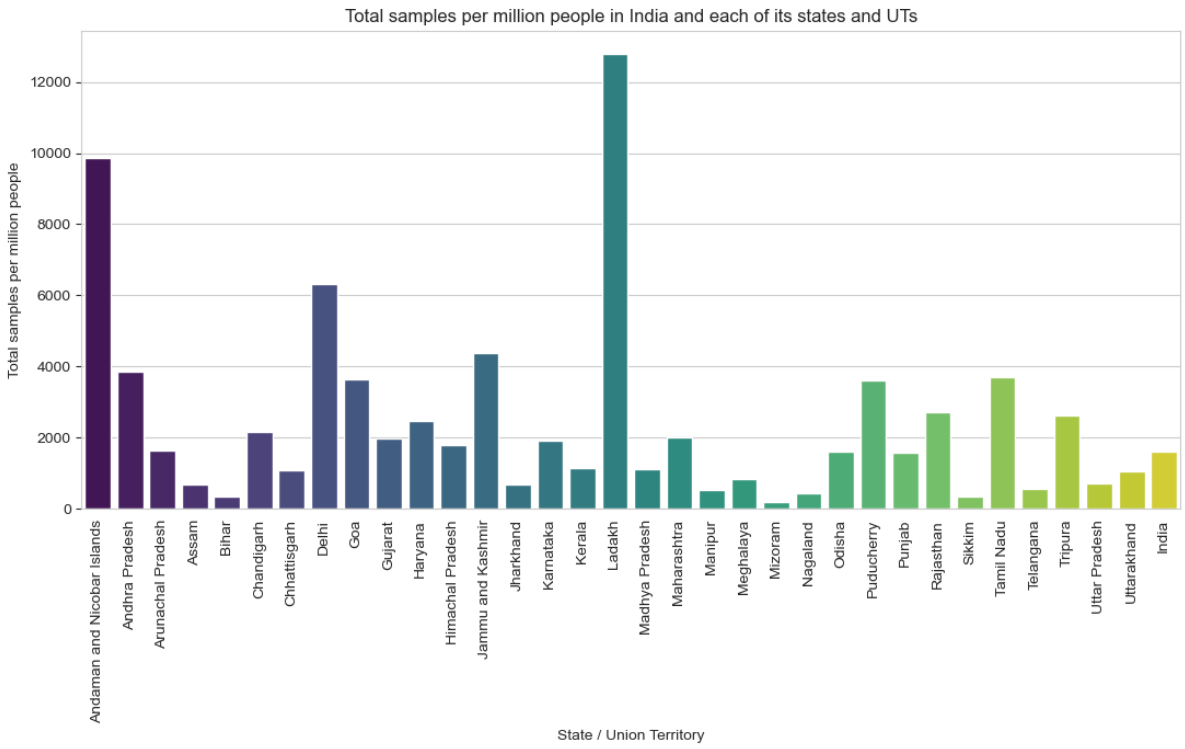
Out[56]:

| | State / Union Territory | TotalSamples | Positive | Population | Total samples per million people | Positive cases per 1000 samples |
|---|---|---|---|---|---|---|
| **0** | Andaman and Nicobar Islands | 3754.0 | 33.0 | 380581 | 9863.866036 | 8.790623 |
| **1** | Andhra Pradesh | 191874.0 | 2051.0 | 49577103 | 3870.214038 | 10.689307 |
| **2** | Arunachal Pradesh | 2257.0 | 2.0 | 1383727 | 1631.102089 | 0.886132 |
| **3** | Assam | 21791.0 | 64.0 | 31205576 | 698.304688 | 2.936992 |
| **4** | Bihar | 37430.0 | 796.0 | 104099452 | 359.560010 | 21.266364 |

# 13.2 Total samples per million people(plot)

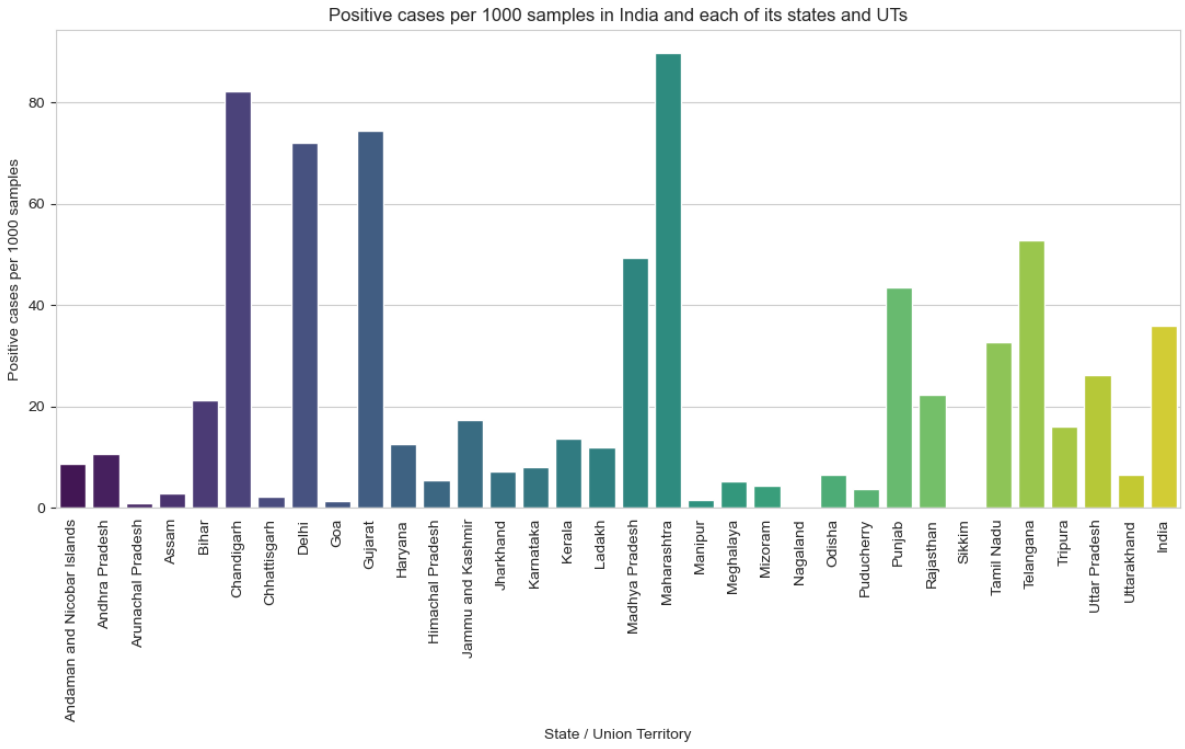
In [57]:
```
plt.figure(figsize=(11,7), dpi = 100)
sns.barplot(x = 'State / Union Territory', y = 'Total samples per million people',
plt.title('Total samples per million people in India and each of its states and UTs
```

```
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```




# 13.3 Positive cases per 1000 samples (plot)

In [58]:
```
plt.figure(figsize=(11, 7), dpi =100)
sns.barplot(x = 'State / Union Territory', y = 'Positive cases per 1000 samples', 
plt.title('Positive cases per 1000 samples in India and each of its states and UTs
plt.xticks(rotation =90)
plt.tight_layout()
plt.show()
```



# Important Covid-19 details of India and each of its states and union territories

In [59]:
```
# create a dataframe with important COVID-19 details for each Indian state/UT as we

# sort by total number of confirmed cases in non-ascending order
covid_19_details = statewise_data.sort_values('Confirmed', ascending = False)

# add serial number (as index) to the dataframe
covid_19_details['Sno'] = range(len(covid_19_details['State / Union Territory']))
covid_19_details.set_index('Sno', inplace = True)

# rearrange the columns in the dataframe
covid_19_details = covid_19_details[['State / Union Territory', 'Confirmed', 'Cure
                                     'Total cases per million people', 'Deaths per

print('IMPORTANT COVID-19 DETAILS OF INDIA AND EACH OF ITS STATES AND UNION TERRITO
covid_19_details
```

```
IMPORTANT COVID-19 DETAILS OF INDIA AND EACH OF ITS STATES AND UNION TERRITORIES
```

Out[59]:

| Sno | State / Union Territory | Confirmed | Cured | Deaths | Active | Recovery rate(%) | Fatality rate(%) | Total cases per million people | Deaths per million people |
|---|---|---|---|---|---|---|---|---|---|
| 0 | India | 74404 | 24386 | 2418 | 47600 | 32.775120 | 3.249825 | 61.462052 | 1.997409 |
| 1 | Maharashtra | 24427 | 5125 | 921 | 18381 | 20.980882 | 3.770418 | 217.371702 | 8.195822 |
| 2 | Gujarat | 8903 | 3246 | 537 | 5120 | 36.459620 | 6.031675 | 147.303861 | 8.884890 |
| 3 | Tamil Nadu | 8718 | 2134 | 61 | 6523 | 24.478091 | 0.699702 | 120.836575 | 0.845496 |
| 4 | Delhi | 7639 | 2512 | 86 | 5041 | 32.883885 | 1.125802 | 455.029000 | 5.122725 |
| 5 | Rajasthan | 4126 | 2378 | 117 | 1631 | 57.634513 | 2.835676 | 60.191015 | 1.706822 |
| 6 | Madhya Pradesh | 3986 | 1860 | 225 | 1901 | 46.663322 | 5.644757 | 54.883315 | 3.098030 |
| 7 | Uttar Pradesh | 3664 | 1873 | 82 | 1709 | 51.118996 | 2.237991 | 18.337206 | 0.410385 |
| 8 | West Bengal | 2173 | 612 | 198 | 1363 | 28.163829 | 9.111827 | 23.806885 | 2.169242 |
| 9 | Andhra Pradesh | 2090 | 1056 | 46 | 988 | 50.526316 | 2.200957 | 42.156558 | 0.927848 |
| 10 | Punjab | 1914 | 171 | 32 | 1711 | 8.934169 | 1.671891 | 68.989535 | 1.153430 |
| 11 | Telangana | 1326 | 830 | 32 | 464 | 62.594268 | 2.413273 | 37.881738 | 0.914190 |
| 12 | Jammu and Kashmir | 934 | 455 | 10 | 469 | 48.715203 | 1.070664 | 76.139037 | 0.815193 |
| 13 | Karnataka | 925 | 433 | 31 | 461 | 46.810811 | 3.351351 | 15.140282 | 0.507404 |
| 14 | Bihar | 831 | 383 | 6 | 442 | 46.089049 | 0.722022 | 7.982751 | 0.057637 |
| 15 | Haryana | 780 | 342 | 11 | 427 | 43.846154 | 1.410256 | 30.767456 | 0.433900 |
| 16 | Kerala | 524 | 489 | 4 | 31 | 93.320611 | 0.763359 | 15.685776 | 0.119739 |
| 17 | Odisha | 437 | 116 | 3 | 318 | 26.544622 | 0.686499 | 10.411153 | 0.071472 |
| 18 | Chandigarh | 187 | 28 | 3 | 156 | 14.973262 | 1.604278 | 177.175612 | 2.842390 |
| 19 | Jharkhand | 172 | 79 | 3 | 90 | 45.930233 | 1.744186 | 5.213996 | 0.090942 |
| 20 | Tripura | 154 | 2 | 0 | 152 | 1.298701 | 0.000000 | 41.917115 | 0.000000 |
| 21 | Uttarakhand | 69 | 46 | 1 | 22 | 66.666667 | 1.449275 | 6.840968 | 0.099144 |
| 22 | Assam | 65 | 39 | 2 | 24 | 60.000000 | 3.076923 | 2.082961 | 0.064091 |
| 23 | Himachal Pradesh | 65 | 39 | 2 | 24 | 60.000000 | 3.076923 | 9.468867 | 0.291350 |
| 24 | Chhattisgarh | 59 | 54 | 0 | 5 | 91.525424 | 0.000000 | 2.309632 | 0.000000 |
| 25 | Ladakh | 42 | 21 | 0 | 21 | 50.000000 | 0.000000 | 153.284672 | 0.000000 |
| 26 | Andaman and Nicobar Islands | 33 | 33 | 0 | 0 | 100.000000 | 0.000000 | 86.709531 | 0.000000 |
| 27 | Puducherry | 13 | 9 | 1 | 3 | 69.230769 | 7.692308 | 10.417059 | 0.801312 |
| 28 | Meghalaya | 13 | 10 | 1 | 2 | 76.923077 | 7.692308 | 4.381694 | 0.337053 |

| Sno | State / Union Territory | Confirmed | Cured | Deaths | Active | Recovery rate(%) | Fatality rate(%) | Total cases per million people | Deaths per million people |
|---|---|---|---|---|---|---|---|---|---|
| **29** | Goa | 7 | 7 | 0 | 0 | 100.000000 | 0.000000 | 4.799303 | 0.000000 |
| **30** | Manipur | 2 | 2 | 0 | 0 | 100.000000 | 0.000000 | 0.778092 | 0.000000 |
| **31** | Arunachal Pradesh | 1 | 1 | 0 | 0 | 100.000000 | 0.000000 | 0.722686 | 0.000000 |
| **32** | Nagaland | 1 | 0 | 0 | 1 | 0.000000 | 0.000000 | 0.505433 | 0.000000 |
| **33** | Mizoram | 1 | 1 | 0 | 0 | 100.000000 | 0.000000 | 0.911406 | 0.000000 |

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: