In [44]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [45]:
```
df = pd.read_csv("covid19_Confirmed_dataset.csv")
```

In [46]:
```
df.head()
```

Out[46]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0

5 rows × 104 columns

In [47]:
```
df.shape
```

Out[47]:
```
(266, 104)
```

In [48]:
```
df.drop(["Lat","Long"],axis=1,inplace=True)
```

In [52]:
```
df.drop(['1/22/20','1/23/20','1/24/20','1/25/20','1/26/20'],axis=1,inplace=Tru
```

In [53]:
```
df.isnull().sum()
```

Out[53]:
```
Province/State    184
Country/Region      0
1/27/20             0
1/28/20             0
1/29/20             0
                  ...
4/26/20             0
4/27/20             0
4/28/20             0
4/29/20             0
4/30/20             0
Length: 97, dtype: int64
```

In [ ]:
```
#merge data country wise
```

In [55]:
```
data_df = df.groupby('Country/Region').sum()
```

In [56]:
```
data_df
```

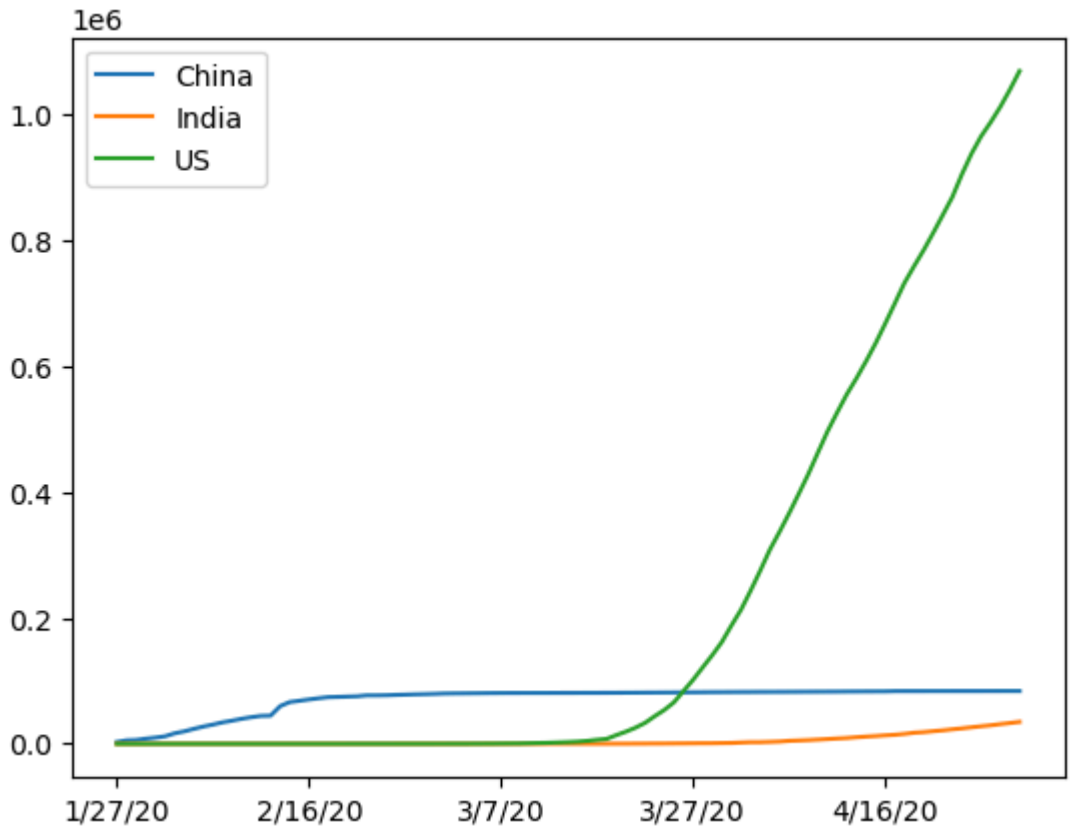
Out[56]:

Country/Region	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/3/20	2/4/20	2/5/20	.
Afghanistan	0	0	0	0	0	0	0	0	0	0	.
Albania	0	0	0	0	0	0	0	0	0	0	.
Algeria	0	0	0	0	0	0	0	0	0	0	.
Andorra	0	0	0	0	0	0	0	0	0	0	.
Angola	0	0	0	0	0	0	0	0	0	0	.
...	...	...	...	...	...	...	...	...	...	...	.
West Bank and Gaza	0	0	0	0	0	0	0	0	0	0	.
Western Sahara	0	0	0	0	0	0	0	0	0	0	.
Yemen	0	0	0	0	0	0	0	0	0	0	.
Zambia	0	0	0	0	0	0	0	0	0	0	.
Zimbabwe	0	0	0	0	0	0	0	0	0	0	.

187 rows × 95 columns

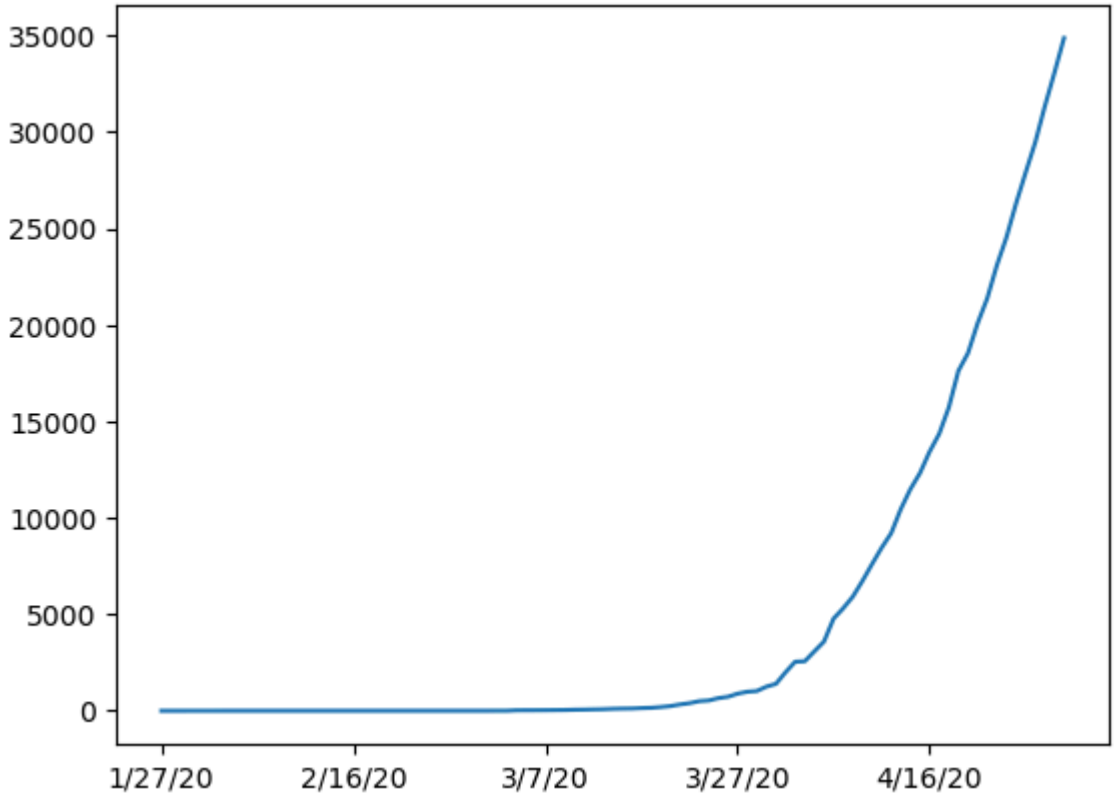
In [65]:
```
data_df.loc["China"].plot()
data_df.loc["India"].plot()
data_df.loc["US"].plot()
plt.legend()
```

Out[65]: <matplotlib.legend.Legend at 0x21cc8204e80>



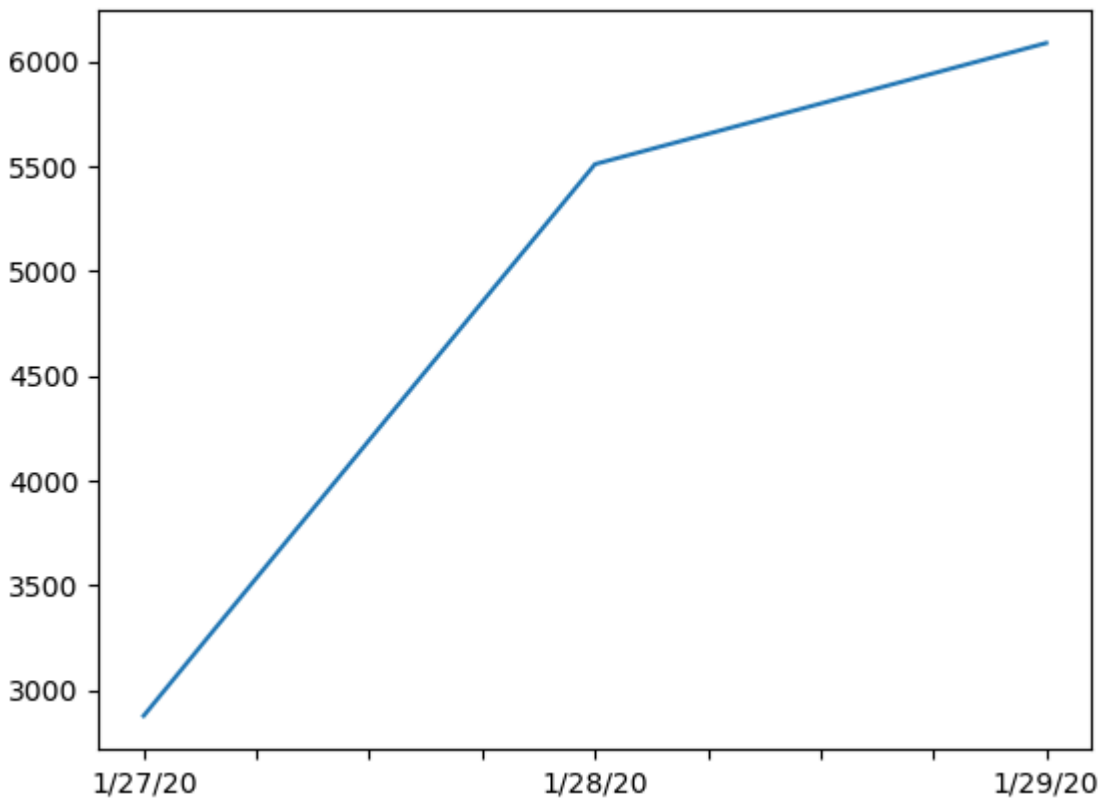
In [66]:
```
data_df.loc['India'].plot()
```

Out[66]: <AxesSubplot:>



In [68]:
```
data_df.loc['China'][:3].plot()
```

Out[68]: <AxesSubplot:>



In [74]:
```
countries=list(data_df.index)
max_infection_rates=[]
for c in countries:
    max_infection_rates.append(data_df.loc[c].diff().max())
data_df["max_infection_rates"]=max_infection_rates
```

In [75]:
```
data_df.head()
```

Out[75]:

Country/Region	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/3/20	2/4/20	2/5/20	.
Afghanistan	0	0	0	0	0	0	0	0	0	0	.
Albania	0	0	0	0	0	0	0	0	0	0	.
Algeria	0	0	0	0	0	0	0	0	0	0	.
Andorra	0	0	0	0	0	0	0	0	0	0	.
Angola	0	0	0	0	0	0	0	0	0	0	.

5 rows × 96 columns

In [77]:
```
data_df=pd.DataFrame(data_df["max_infection_rates"])
```

In [81]:
```
happy_df=pd.read_csv("worldwide_happiness_report.csv")
```

In [82]:
```
happy_df.shape
```

Out[82]:
```
(156, 9)
```

In [83]:
```
happy_df.head()
```

Out[83]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298

In [92]:
```
cols=["Overall rank","Score","Generosity","Perceptions of corruption"]
```

In [93]:
```
happy_df.drop(cols,axis=1,inplace=True)
happy_df.head()
```

Out[93]:

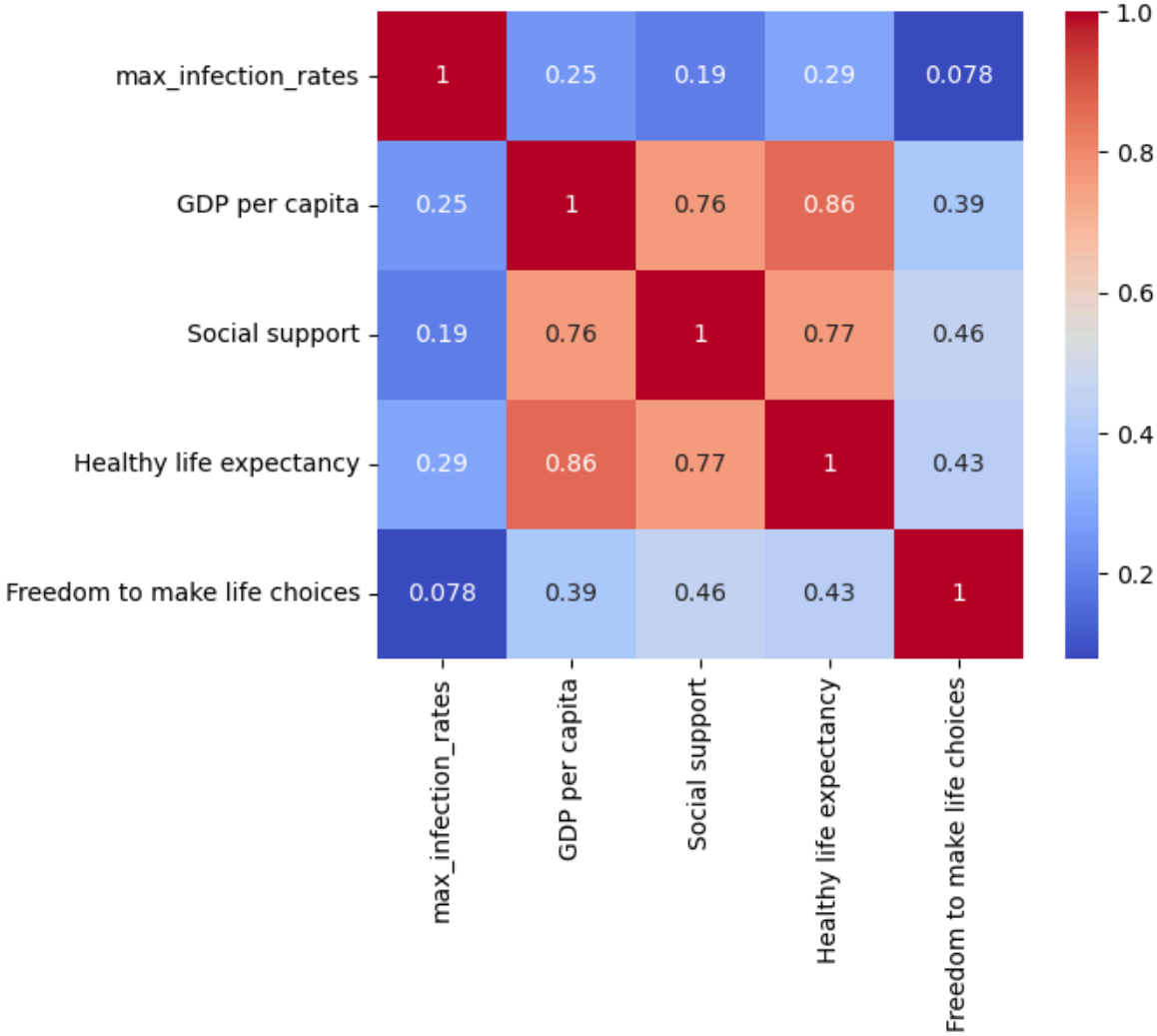
	Country or region	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
0	Finland	1.340	1.587	0.986	0.596
1	Denmark	1.383	1.573	0.996	0.592
2	Norway	1.488	1.582	1.028	0.603
3	Iceland	1.380	1.624	1.026	0.591
4	Netherlands	1.396	1.522	0.999	0.557

In [95]:
```
happy_df.set_index("Country or region",inplace=True)
```

In [98]:
```
final=data_df.join(happy_df,how="inner")
```

In [114]:
```
corr_matrix=final.corr()
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", square=True)
```

Out[114]: <AxesSubplot:>

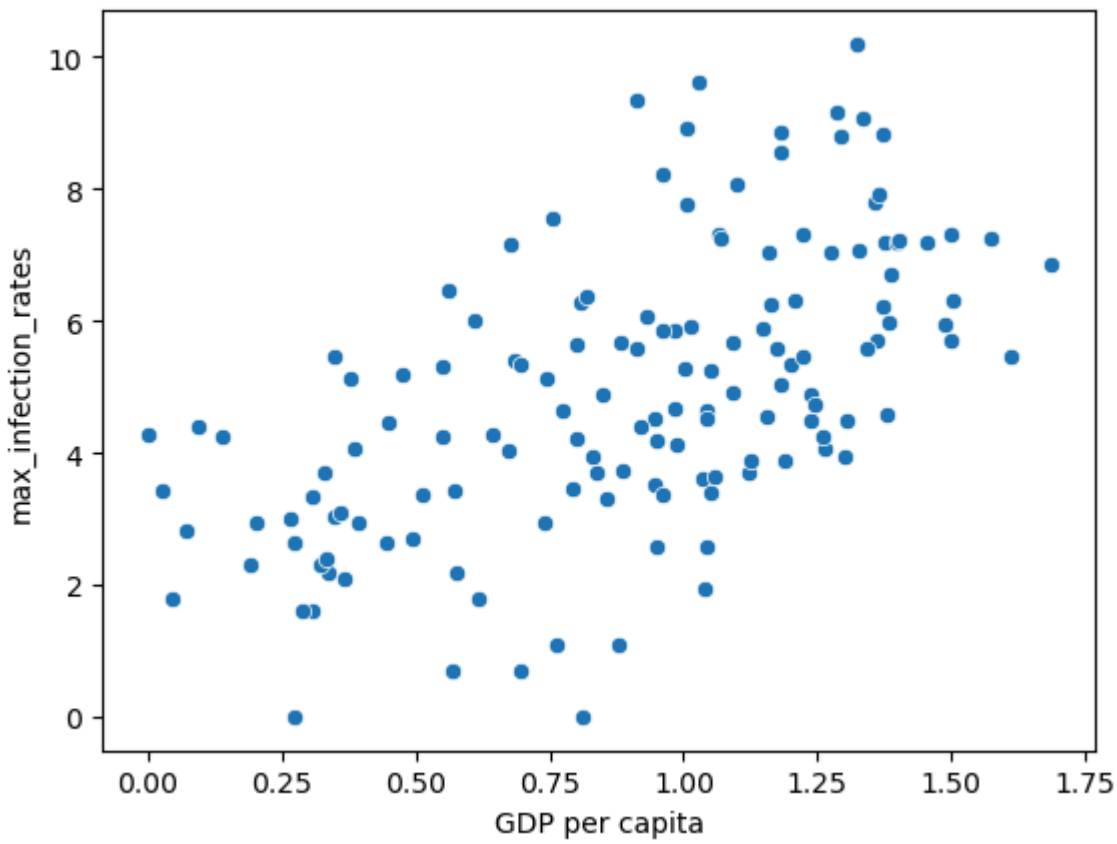


In [102]:
```
#plot gdp vs maximum infection rate
```

In [103]:
```
x=final["GDP per capita"]
y=final["max_infection_rates"]
sns.scatterplot(x,np.log(y))
```

```
C:\Users\Tejaswini\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fu
tureWarning: Pass the following variables as keyword args: x, y. From versio
n 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpre
tation.
  warnings.warn(
```

Out[103]:
```
<AxesSubplot:xlabel='GDP per capita', ylabel='max_infection_rates'>
```

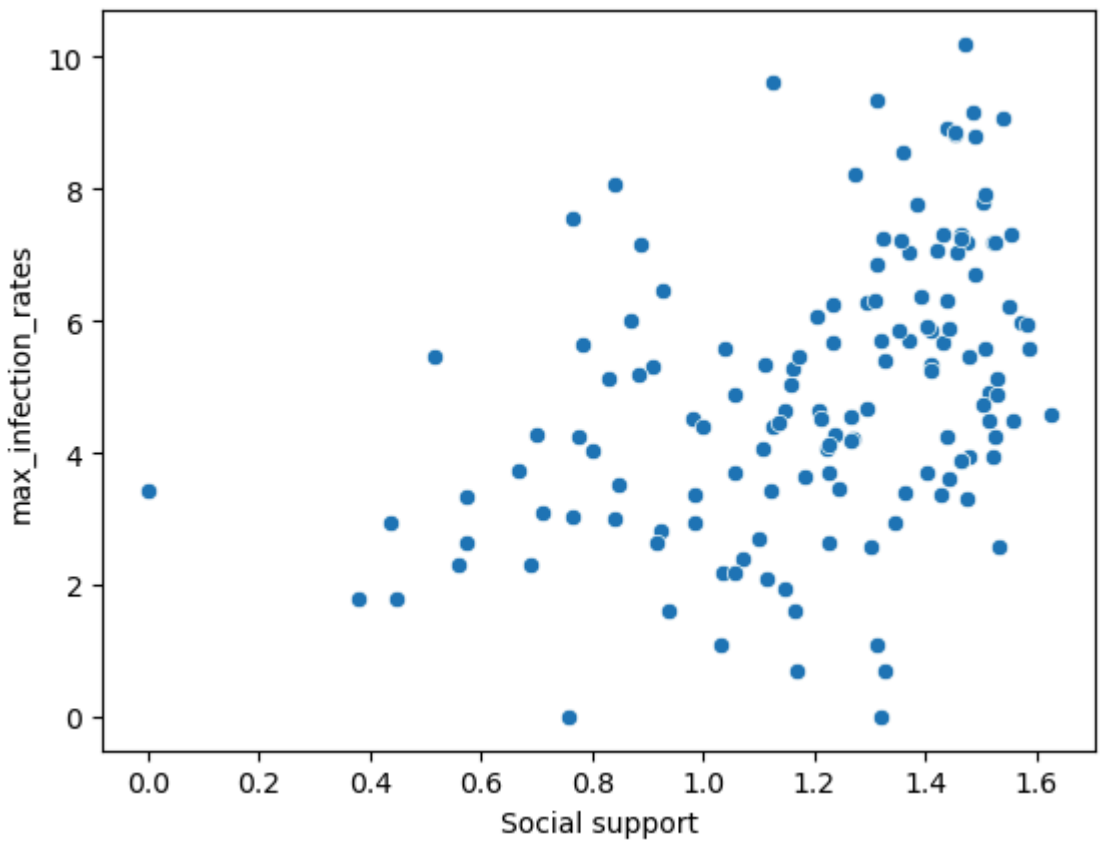


In [104]:
```
#plot social support vs maximum Infection rate
```

In [107]:
```
x=final["Social support"]
y=final["max_infection_rates"]
sns.scatterplot(x,np.log(y))
```

```
C:\Users\Tejaswini\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fu
tureWarning: Pass the following variables as keyword args: x, y. From versio
n 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpre
tation.
  warnings.warn(
```

Out[107]:
```
<AxesSubplot:xlabel='Social support', ylabel='max_infection_rates'>
```

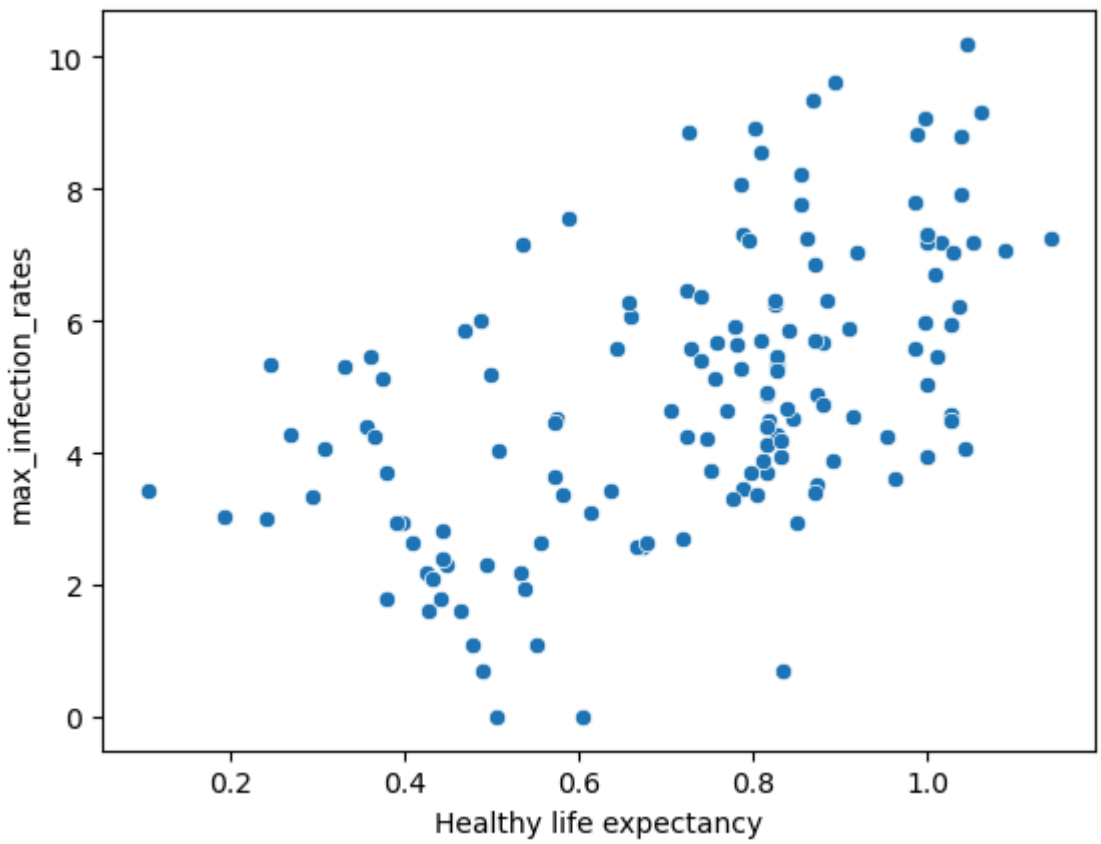


In [108]:
```
#plot Healthy life expectancy vs maximum Infection rate
```

In [109]:
```
x=final["Healthy life expectancy"]
y=final["max_infection_rates"]
sns.scatterplot(x,np.log(y))
```

```
C:\Users\Tejaswini\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fu
tureWarning: Pass the following variables as keyword args: x, y. From versio
n 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpre
tation.
  warnings.warn(
```

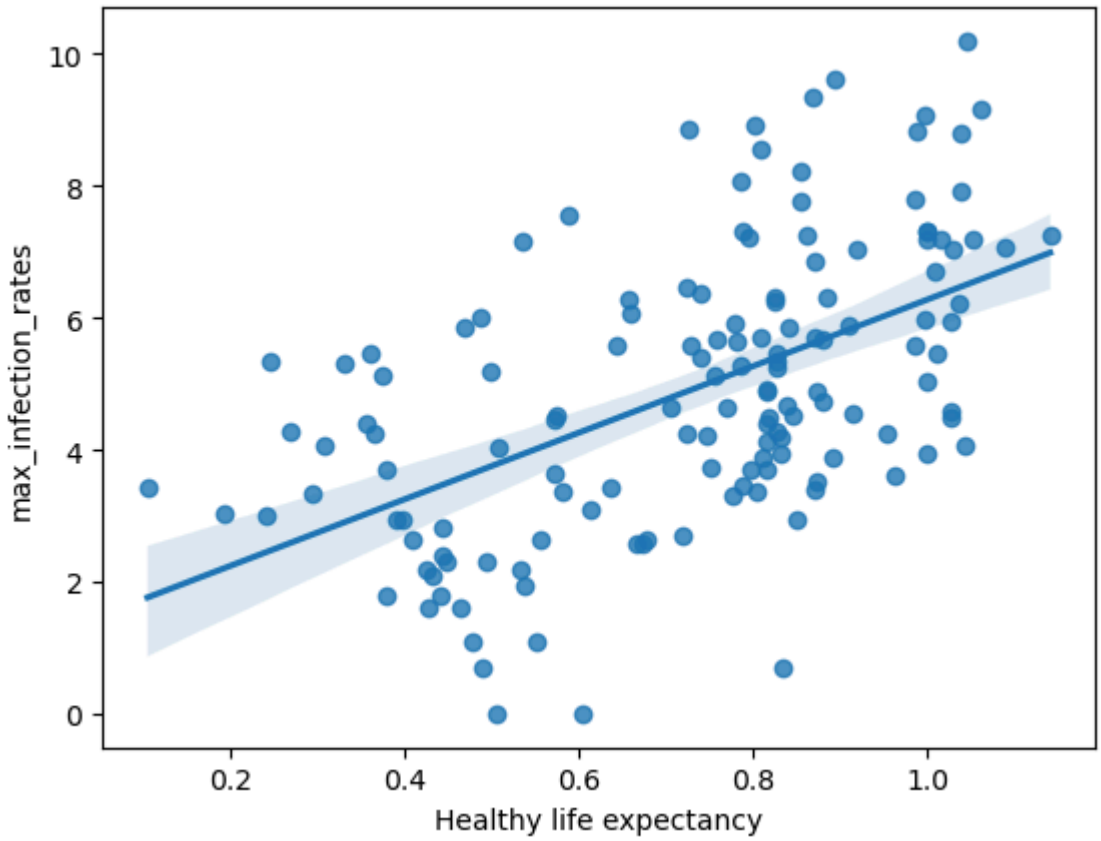
Out[109]:
```
<AxesSubplot:xlabel='Healthy life expectancy', ylabel='max_infection_rates'>
```



In [110]:
```
sns.regplot(x,np.log(y))
```

```
C:\Users\Tejaswini\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fu
tureWarning: Pass the following variables as keyword args: x, y. From versio
n 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpre
tation.
  warnings.warn(
```

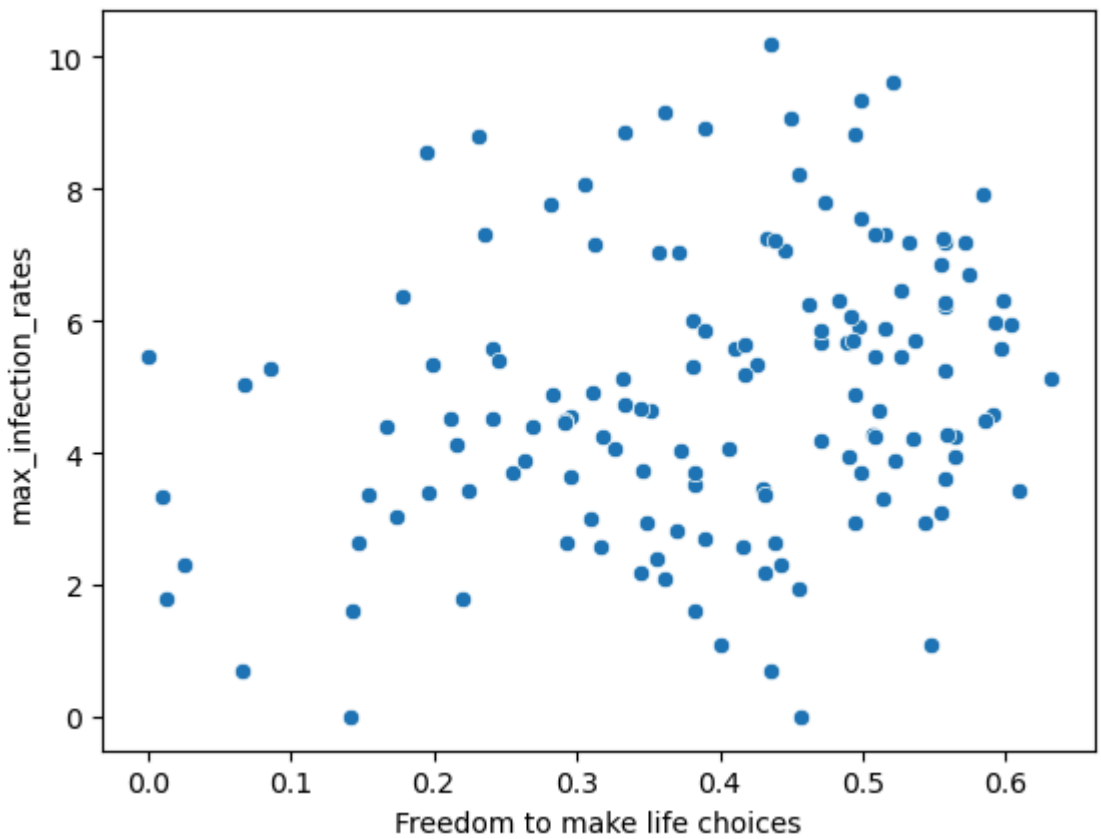
Out[110]:
```
<AxesSubplot:xlabel='Healthy life expectancy', ylabel='max_infection_rates'>
```



In [111]:
```
#Plot Freedom to make life choices vs maximum Infection rate¶
```

In [112]:
```
x=final["Freedom to make life choices"]
y=final["max_infection_rates"]
sns.scatterplot(x,np.log(y))
```

Out[112]:
```
<AxesSubplot:xlabel='Freedom to make life choices', ylabel='max_infection_ra
tes'>
```



In [225]:
```
df_state = pd.read_csv("Total_India_covid-19.csv")
```

In [226]:
```
df_state.shape
```

Out[226]:
```
(37, 9)
```

In [227]:
```
df_state.head()
```

Out[227]:

	State	Statecode	Confirmed	Active	Recovered	Deaths	Last Updated	Latitude	Longitu
0	Maharashtra	MH	375799	148601	213238	13656	26/07/2020 20:06:27	19.7515	75.71
1	Tamil Nadu	TN	213723	53703	156526	3494	26/07/2020 18:17:27	11.1271	78.65
2	Delhi	DL	130606	11904	114875	3827	26/07/2020 15:08:29	28.7041	77.10
3	Karnataka	KA	96141	58414	35838	1880	27/07/2020 00:56:29	15.3173	75.71
4	Andhra Pradesh	AP	96298	48956	46301	1041	26/07/2020 18:26:30	15.9129	79.74

In [228]:
```
#drop unnecessary columns
```

In [229]:
```
df_state.drop(['Latitude','Longitude'],axis=1,inplace=True)
```

In [230]:
```
df_state['Last Updated'] = pd.to_datetime(df_state['Last Updated'], format='%d
```

In [231]:
```
df_state['year'] = df_state['Last Updated'].dt.year
```

In [232]:
```
df_state.head()
```

Out[232]:

	State	Statecode	Confirmed	Active	Recovered	Deaths	Last Updated	year
0	Maharashtra	MH	375799	148601	213238	13656	2020-07-26 20:06:27	2020
1	Tamil Nadu	TN	213723	53703	156526	3494	2020-07-26 18:17:27	2020
2	Delhi	DL	130606	11904	114875	3827	2020-07-26 15:08:29	2020
3	Karnataka	KA	96141	58414	35838	1880	2020-07-27 00:56:29	2020
4	Andhra Pradesh	AP	96298	48956	46301	1041	2020-07-26 18:26:30	2020

In [233]:
```
df_state.drop(['Last Updated'],axis=1,inplace=True)
```

In [234]:
```
df_state.head()
```

Out[234]:

	State	Statecode	Confirmed	Active	Recovered	Deaths	year
0	Maharashtra	MH	375799	148601	213238	13656	2020
1	Tamil Nadu	TN	213723	53703	156526	3494	2020
2	Delhi	DL	130606	11904	114875	3827	2020
3	Karnataka	KA	96141	58414	35838	1880	2020
4	Andhra Pradesh	AP	96298	48956	46301	1041	2020

In [235]:
```
#let plot the data
```

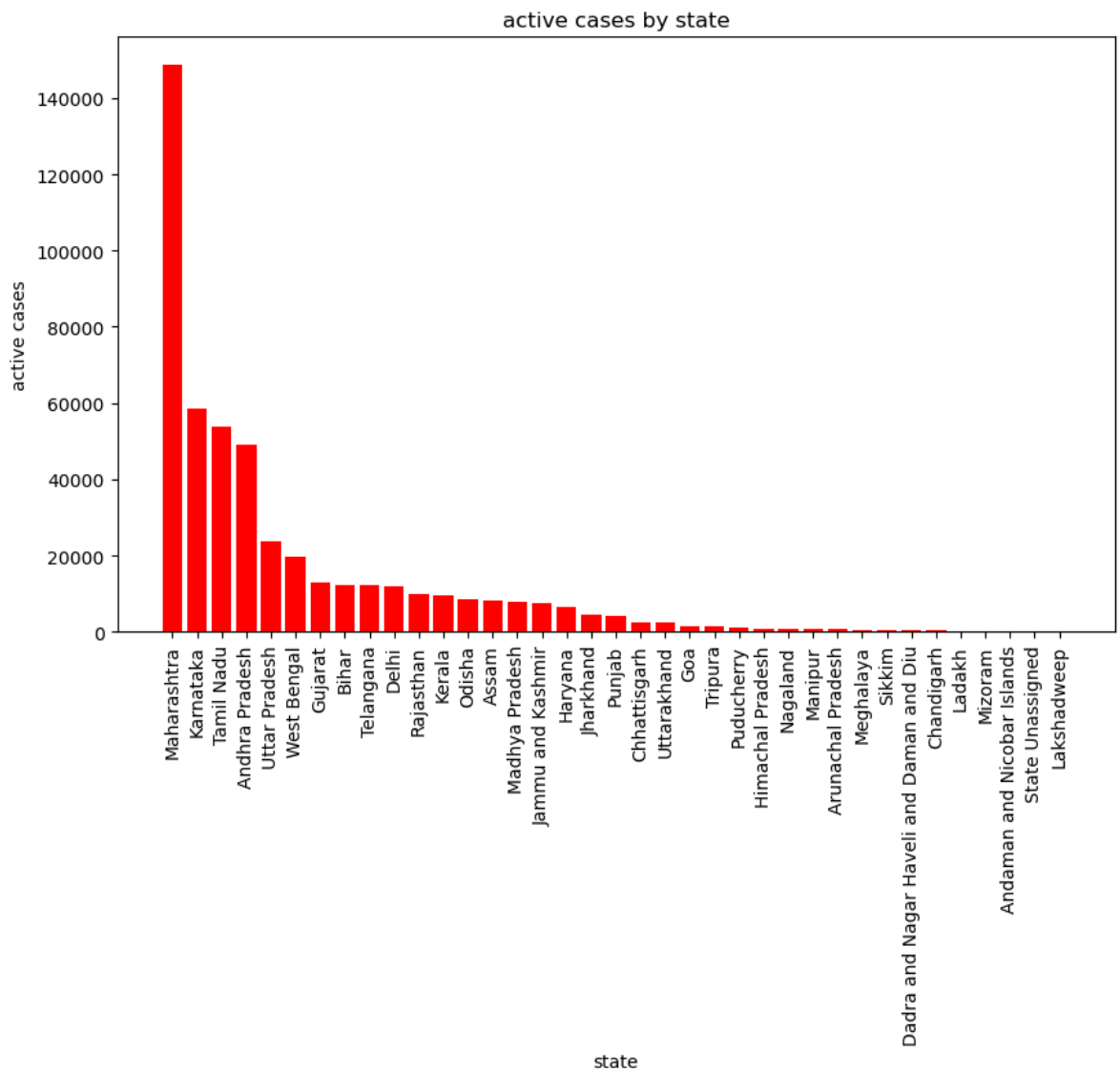
In [236]:
```
df_state_sorted = df_state.sort_values(by='Active',ascending=False)
df_state_sorted
```

Out[236]:

	State	Statecode	Confirmed	Active	Recovered	Deaths	year
0	Maharashtra	MH	375799	148601	213238	13656	2020
3	Karnataka	KA	96141	58414	35838	1880	2020
1	Tamil Nadu	TN	213723	53703	156526	3494	2020
4	Andhra Pradesh	AP	96298	48956	46301	1041	2020
5	Uttar Pradesh	UP	66988	23921	41641	1426	2020
7	West Bengal	WB	58718	19595	37751	1372	2020
6	Gujarat	GJ	55822	13033	40467	2322	2020
10	Bihar	BR	38919	12361	26308	249	2020
8	Telangana	TG	54059	12264	41332	463	2020
2	Delhi	DL	130606	11904	114875	3827	2020
9	Rajasthan	RJ	36430	9852	25954	624	2020
16	Kerala	KL	19026	9655	9300	62	2020
14	Odisha	OR	25389	8422	16793	174	2020
12	Assam	AS	32229	8106	24041	79	2020
13	Madhya Pradesh	MP	27800	7857	19132	811	2020
15	Jammu and Kashmir	JK	17920	7680	9928	312	2020
11	Haryana	HR	31332	6556	24384	392	2020
18	Jharkhand	JH	8349	4562	3704	83	2020
17	Punjab	PB	13218	4102	8810	306	2020
19	Chhattisgarh	CT	7613	2626	4944	43	2020
20	Uttarakhand	UT	6104	2437	3566	63	2020
21	Goa	GA	4861	1549	3277	35	2020
22	Tripura	TR	3919	1526	2362	13	2020
23	Puducherry	PY	2787	1102	1645	40	2020
25	Himachal Pradesh	HP	2176	950	1198	13	2020
27	Nagaland	NL	1339	786	549	4	2020
24	Manipur	MN	2235	714	1521	0	2020
28	Arunachal Pradesh	AR	1158	650	505	3	2020
31	Meghalaya	ML	702	562	135	5	2020
32	Sikkim	SK	558	397	147	1	2020
30	Dadra and Nagar Haveli and Daman and Diu	DN	950	373	565	2	2020
29	Chandigarh	CH	887	302	572	13	2020
26	Ladakh	LA	1285	218	1063	4	2020
33	Mizoram	MZ	361	178	183	0	2020
34	Andaman and Nicobar Islands	AN	324	141	182	0	2020

	State	Statecode	Confirmed	Active	Recovered	Deaths	year
35	State Unassigned	UN	0	0	0	0	2020
36	Lakshadweep	LD	0	0	0	0	2020

In [237]:
```
plt.figure(figsize=(10,6))
plt.bar(df_state_sorted['State'],df_state_sorted['Active'],color='red')
plt.xlabel('state')
plt.ylabel('active cases')
plt.title('active cases by state')
plt.xticks(rotation=90)
plt.show()
```



In [238]:
```
#what is year analysis for maharastra
```

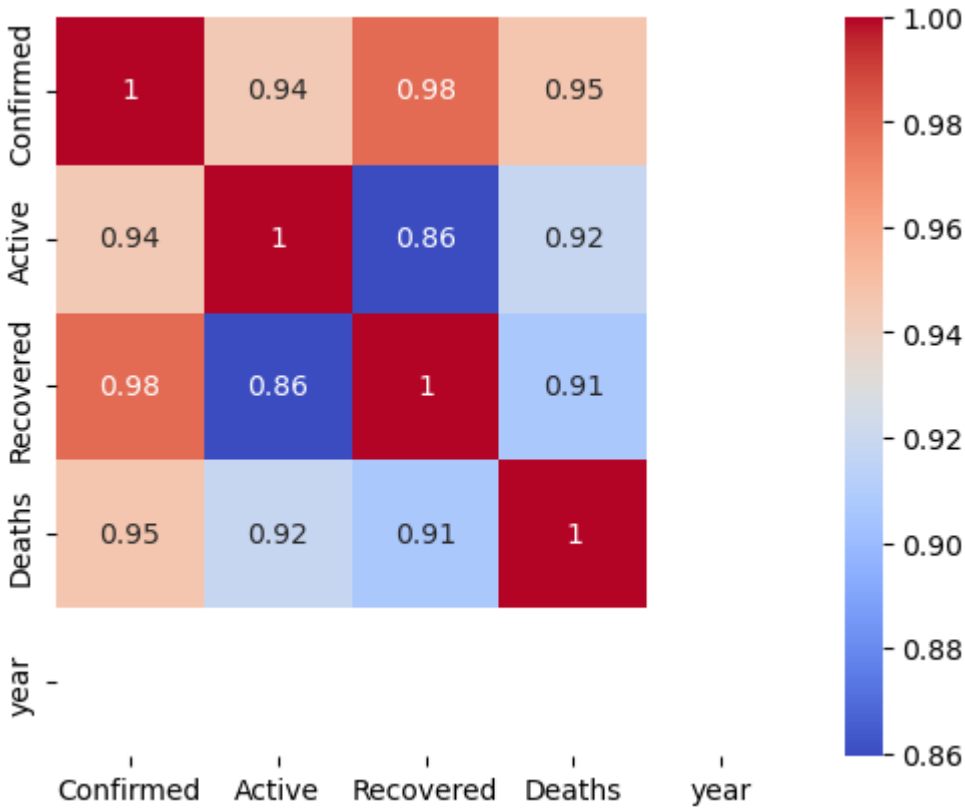
In [ ]:

In [207]:
```
df_state_sorted = df_state_sorted['year'] .unique()
df_state_sorted
```

Out[207]:
```
array([2020], dtype=int64)
```

In [219]:
```
#df_state_sorted_corr = df_state_sorted.corr()
#sns.heatmap(df_state_sorted_corr,annot=True,cmap='coolwarm',square=True)
```

Out[219]:
```
<AxesSubplot:>
```

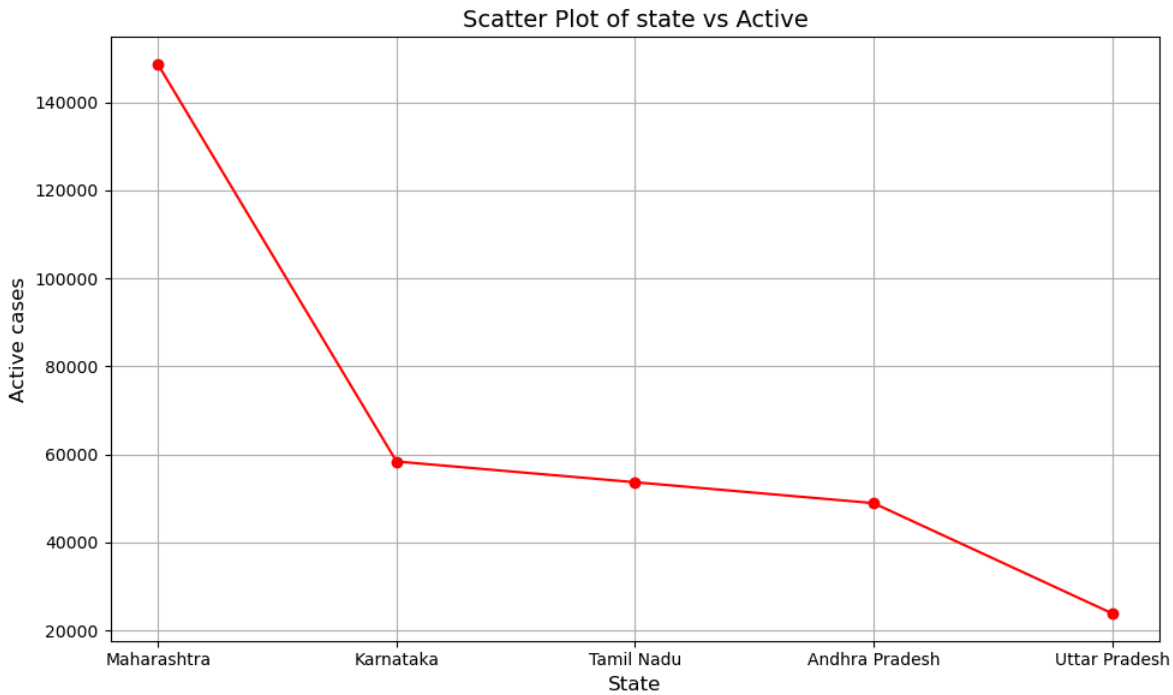


In [242]:
```
df_statnew_states = df_state_sorted["State"][:5]
```

In [252]:
```
plt.figure(figsize=(10, 6))  # Adjust figure size as needed

# Scatter plot for two columns
plt.scatter(df_state_sorted['State'][:5], df_state_sorted['Active'][:5], color

plt.xlabel('State', fontsize=12)  # Customize the x-axis label
plt.plot(df_state_sorted['State'][:5],df_state_sorted['Active'][:5],color='red
plt.ylabel('Active cases', fontsize=12)  # Customize the y-axis label
plt.title('Scatter Plot of state vs Active', fontsize=14)  # Customize the tit
plt.grid(True)  # Add gridlines
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```

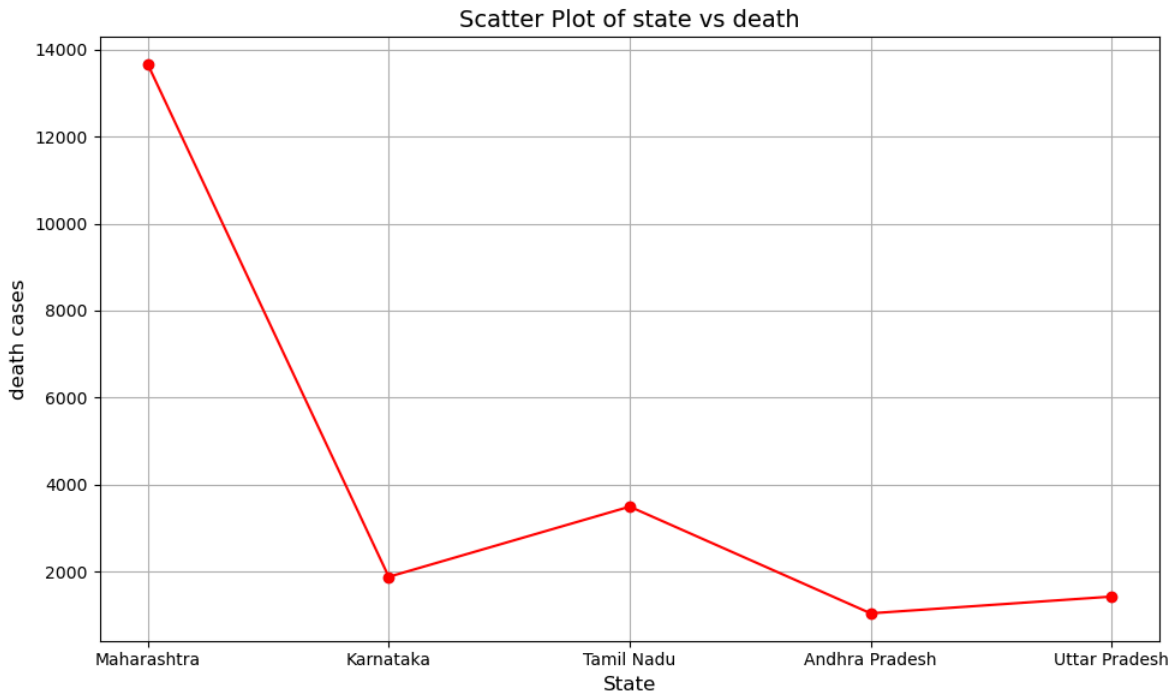


In [ ]:
```
#The active more in maharastra,karnatak,tamilnadu in first three places
```

In [253]:
```
plt.figure(figsize=(10, 6))  # Adjust figure size as needed

# Scatter plot for two columns
plt.scatter(df_state_sorted['State'][:5], df_state_sorted['Deaths'][:5], color

plt.xlabel('State', fontsize=12)  # Customize the x-axis label
plt.ylabel('death cases', fontsize=12)  # Customize the y-axis label
plt.plot(df_state_sorted['State'][:5],df_state_sorted['Deaths'][:5],color='red
plt.title('Scatter Plot of state vs death', fontsize=14)  # Customize the titl
plt.grid(True)  # Add gridlines
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```



In [254]:
```
#The deaths more in maharastra,Tamilnadu,karanataka in first three places. And
#In maharastra 14000 was active cases and close to that was die. It says that
#That may be tells that Andhra pradesh has good facilities and vaccine centres
#may be Andhra prople has more resistance power
```

In [ ]: