Visualize current state - plot COVID data

In [1]:

```
countryToAnalyze = "India"
stateToAnalyze ="Karnataka"
```

In [2]:

```
#Download data from "https://github.com/CSSEGISandData/COVID-19.git"

confirmedCsv = "COVID-19/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Confirmed.csv"
recoveredCsv = "COVID-19/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Recovered.csv"
deathsCsv = "COVID-19/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Deaths.csv"

try:
    f = open(confirmedCsv)
except IOError:
    print('Download data from "https://github.com/CSSEGISandData/COVID-19.git"')
    assert False
finally:
    f.close()
```

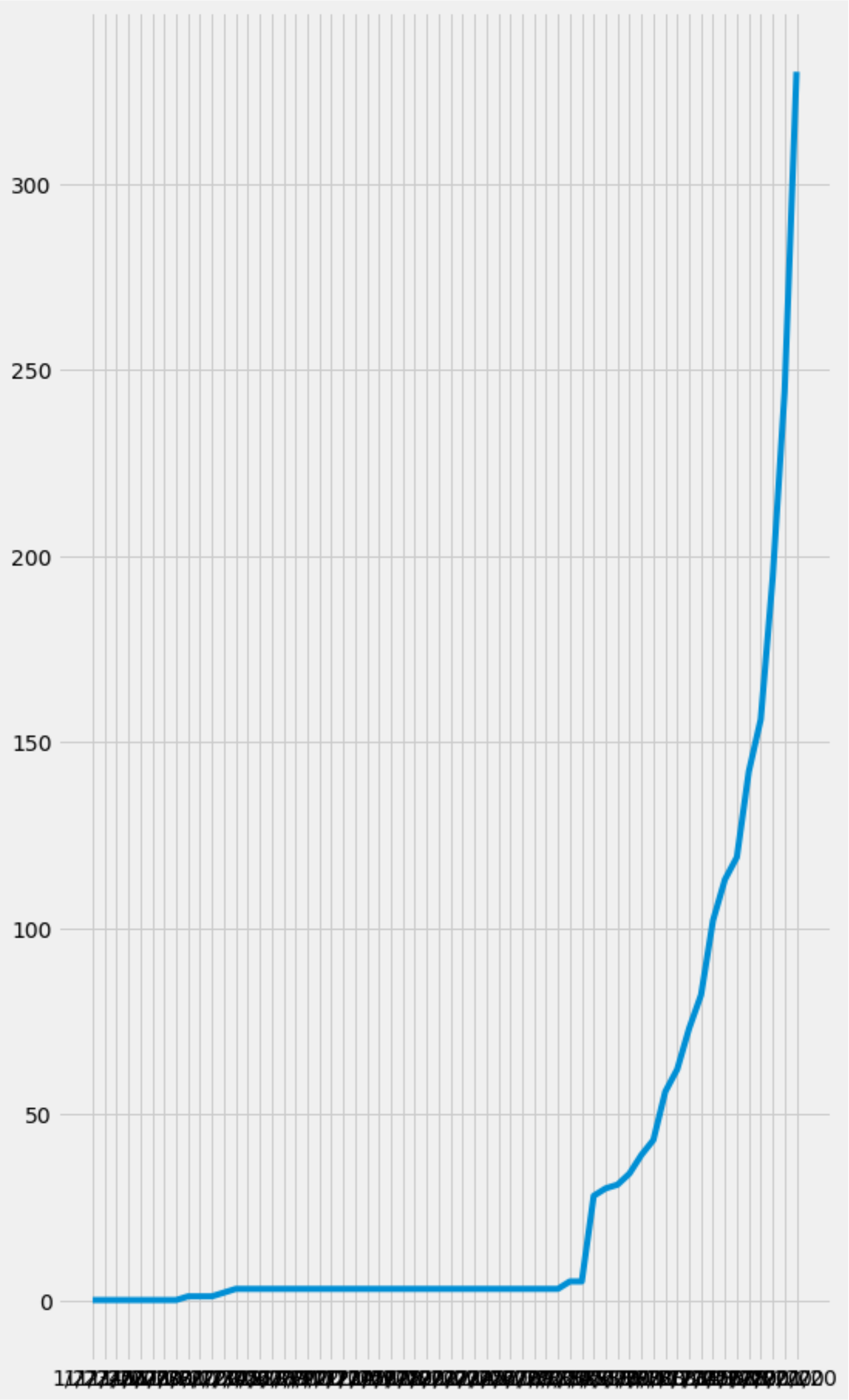
In [3]:

```
import pandas as pd

confirmedDf = pd.read_csv(confirmedCsv)
recoveredDf = pd.read_csv(recoveredCsv)
deathsDf = pd.read_csv(deathsCsv)
```

In [4]:

```
from matplotlib import pyplot
%matplotlib inline

pyplot.style.use("fivethirtyeight")# for pretty graphs

# Increase the default plot size and set the color scheme
pyplot.rcParams['figure.figsize'] = 8, 15

confirmedTSDf = confirmedDf.loc[confirmedDf["Country/Region"] == countryToAnalyze].T[4:]
pyplot.figure(1)
pyplot.plot(confirmedTSDf)
```

```
Out[4]:

[<matplotlib.lines.Line2D at 0x11997bdd0>]
```

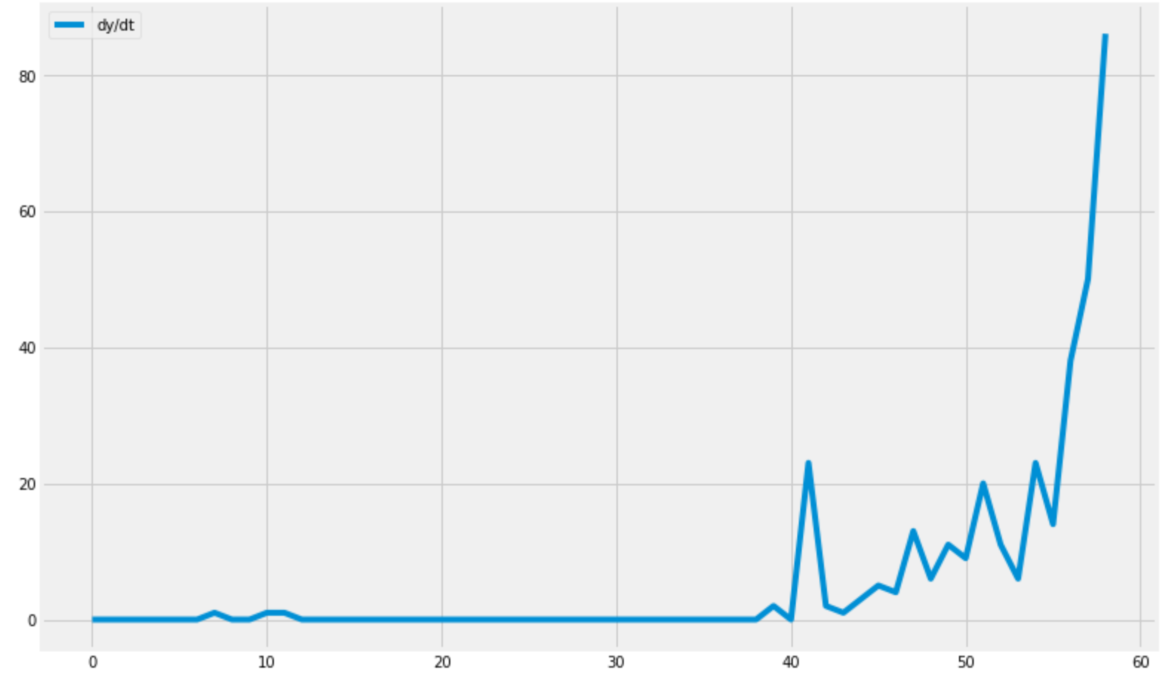

300
250
200
150
100
50
0

In [5]:

```
%matplotlib inline
import numpy as np
dy_dt= np.diff(confirmedTSDf[15].values)
f, ax = pyplot.subplots(figsize=(12, 8))
pyplot.plot(dy_dt, label="dy/dt")
pyplot.legend()
```

Out[5]:

```
<matplotlib.legend.Legend at 0x11a6f4e50>
```

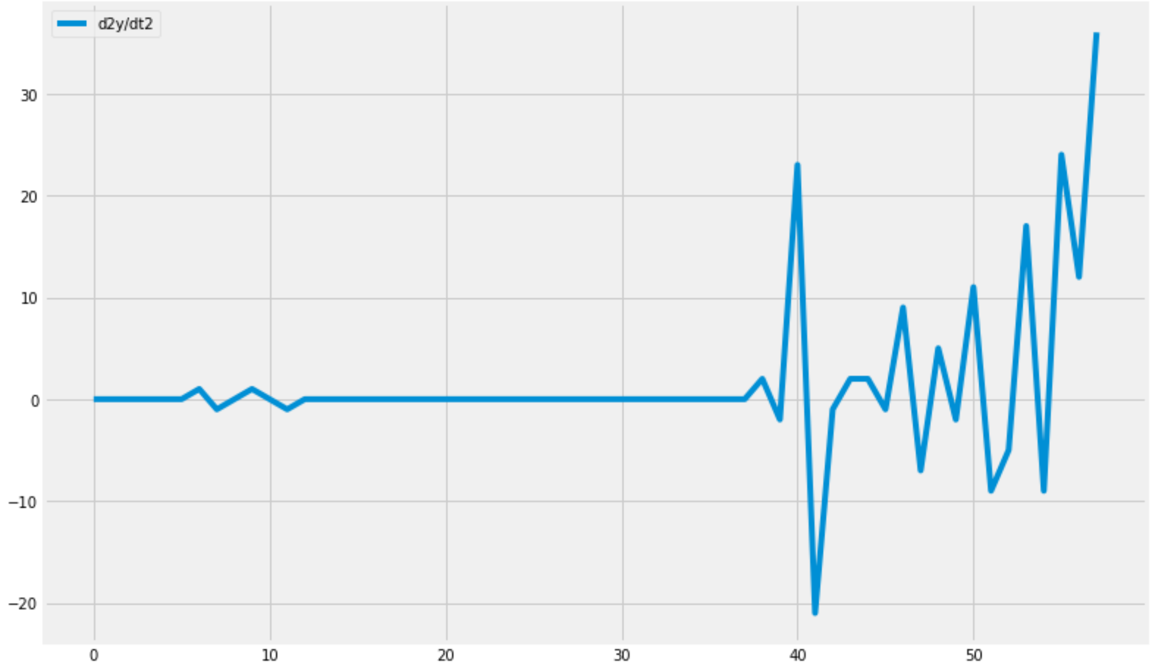


In [6]:

```
d2y_dt2= np.diff(dy_dt)
f, ax = pyplot.subplots(figsize=(12, 8))
pyplot.plot(d2y_dt2, label="d2y/dt2")
pyplot.legend()
```

Out[6]:

```
<matplotlib.legend.Legend at 0x11a9bfb10>
```



Analyze statewise for India

Download data from - [https://www.kaggle.com/sudalairajkumar/covid19-in-india (https://www.kaggle.com/sudalairajkumar/covid19-in-india)](https://www.kaggle.com/sudalairajkumar/covid19-in-india)

In [7]:

```
from datetime import datetime

covidDataFile = "covid19-in-india/covid_19_india.csv"
populationFile = "covid19-in-india/population_india_census2011.csv"
hospitalBedsFile = "covid19-in-india/HospitalBedsIndia.csv"

try:
    f = open(covidDataFile)
except IOError:
    print('Download data from "https://www.kaggle.com/sudalairajkumar/covid19-in
-india"')
    assert False
finally:
    f.close()


def parser(x):
    return datetime.strptime(x, '%d/%m/%y')

covidIndiaDataDf = pd.read_csv(covidDataFile, parse_dates=[1], index_col=1, sque
eze=True, date_parser=parser)
populationDf = pd.read_csv(populationFile)
hospitalBedsDf = pd.read_csv(hospitalBedsFile)
```

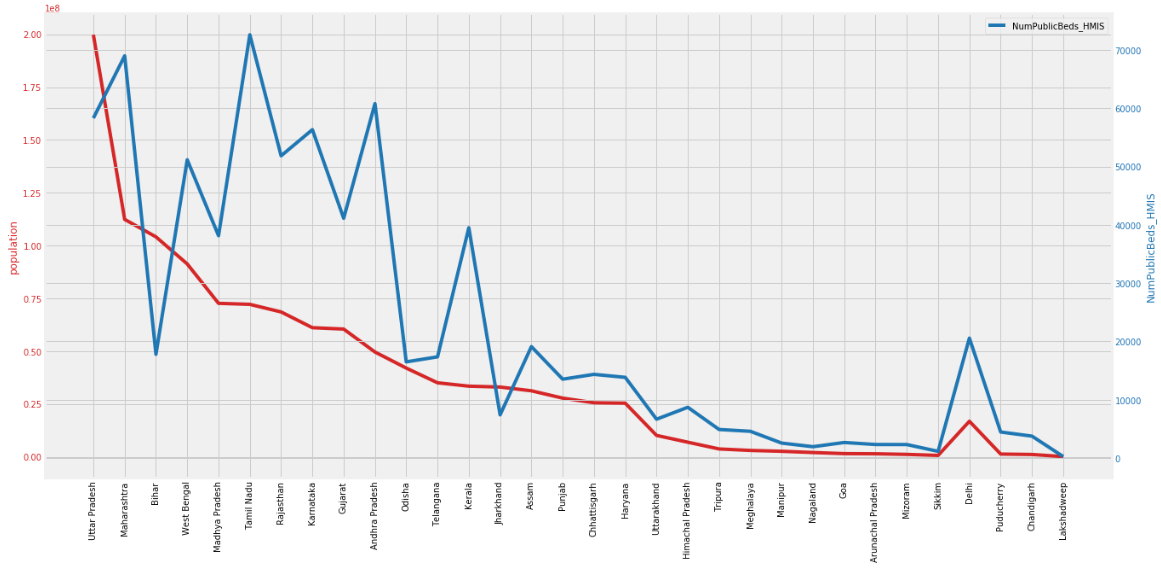
Capacity for maximum

In [8]:

```
%matplotlib inline


hospitalBedsDf.dropna(axis=0, how='all', inplace=True)
hospitalBedsDf = hospitalBedsDf.fillna(-1)
hospitalBedsDf = hospitalBedsDf.loc[hospitalBedsDf["State/UT"] != "All India"]
hospitalBedsDf["State / Union Territory"] = hospitalBedsDf["State/UT"]

populationHospitalBedsdf = pd.merge(populationDf, hospitalBedsDf, on=['State / U
nion Territory'])
states = populationHospitalBedsdf["State / Union Territory"]

fig, ax1 = pyplot.subplots(figsize=(20,10))

color = 'tab:red'
ax1.set_ylabel('population', color=color)
pyplot.xticks(rotation=90)
ax1.plot(states, populationHospitalBedsdf["Population"], label ="population", co
lor=color)
ax1.tick_params(axis='y', labelcolor=color)

color = 'tab:blue'
ax2 = ax1.twinx()
ax2.set_ylabel('NumPublicBeds_HMIS', color=color)

ax2.plot(states, populationHospitalBedsdf["NumPublicBeds_HMIS"].astype(int), lab
el="NumPublicBeds_HMIS", color=color)
ax2.tick_params(axis='y', labelcolor=color)
pyplot.legend()
```

Out[8]:

```
<matplotlib.legend.Legend at 0x11aced990>
```



State-wise numbers

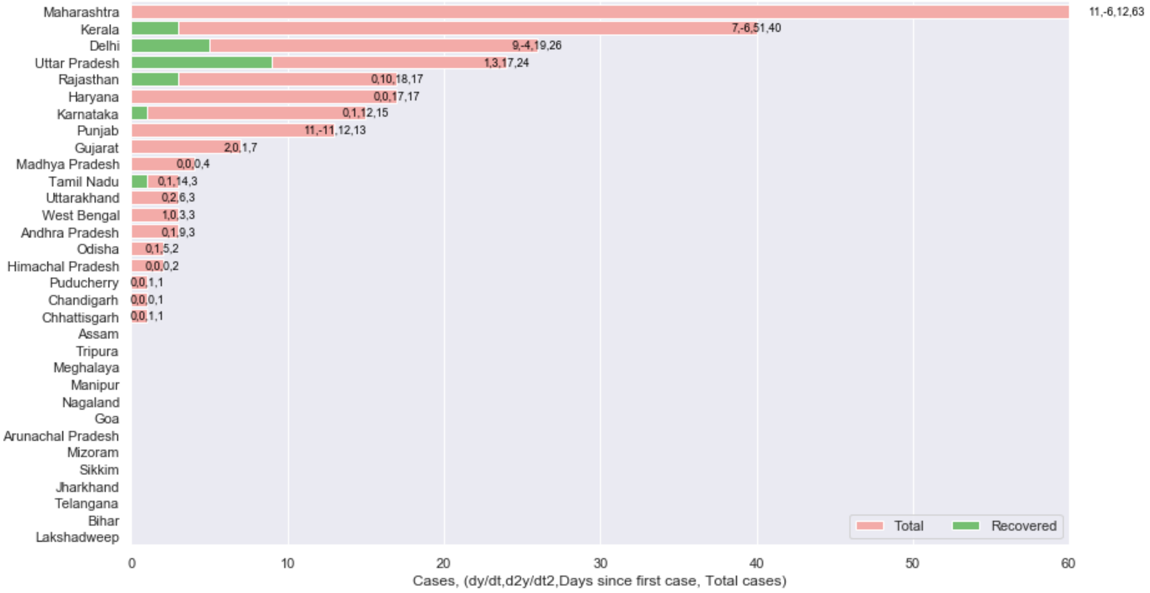
In [9]:

```
covidIndiaLastDayDataDf = pd.DataFrame(columns=covidIndiaDataDf.columns.values)

covidIndiaLastDayDataDf.insert(len(covidIndiaDataDf.columns), "dy_dt", [], True)
covidIndiaLastDayDataDf.insert(len(covidIndiaDataDf.columns), "d2y_dt2", [], True)
covidIndiaLastDayDataDf.insert(len(covidIndiaDataDf.columns), "days", [], True)


for state in states:
    stateDataDf = covidIndiaDataDf.loc[covidIndiaDataDf["State/UnionTerritory"]==state]
    stateDataDf.sort_values('Date',ascending=False,inplace=True)
    if stateDataDf.shape[0] != 0:
        covidIndiaLastDayDataDf = covidIndiaLastDayDataDf.append(stateDataDf.iloc[0])
        dy_dt= np.diff(stateDataDf["ConfirmedIndianNational"].values)
        d2y_dt2= np.diff(dy_dt)
        days = 0
        if len(stateDataDf.index) > 0:
            date = stateDataDf.index[-1]
            days = (stateDataDf.index[0] - stateDataDf.index[-1]).days
            #print(days)
        else:
            date=datetime.strptime("1/1/1970", '%d/%m/%y')
        if date == 0:
            date=datetime.strptime("1/1/1970", '%d/%m/%y')
        last_dy_dt = 0
        if dy_dt.shape[0] != 0:
            last_dy_dt = dy_dt[0]
        last_d2y_dt2 = 0
        if d2y_dt2.shape[0] != 0:
            last_d2y_dt2 = d2y_dt2[0]
        covidIndiaLastDayDataDf.iloc[-1,-1] = -1*last_dy_dt
        covidIndiaLastDayDataDf.iloc[-1, -2] = -1*last_d2y_dt2
        covidIndiaLastDayDataDf.iloc[-1, -3] = days
    else:
        covidIndiaLastDayDataDf = covidIndiaLastDayDataDf.append(pd.Series(), ignore_index=True)
        covidIndiaLastDayDataDf.iloc[-1, 1] = state


covidIndiaLastDayDataDf.fillna(0, inplace=True)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # Remove the CWD from sys.path while we load stuff.
```

In [10]:

```
import seaborn as sns

# Disable warnings
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
sns.set()

f, ax = pyplot.subplots(figsize=(12, 8))
covidIndiaLastDayDataDf['Name of State / UT']=covidIndiaLastDayDataDf['State/Uni
onTerritory']
covidIndiaLastDayDataDf['Total cases']=covidIndiaLastDayDataDf['ConfirmedIndianN
ational']+covidIndiaLastDayDataDf['ConfirmedForeignNational']
covidIndiaLastDayDataDf['Cured/Discharged/Migrated']=covidIndiaLastDayDataDf['Cu
red']
data = covidIndiaLastDayDataDf[['Name of State / UT','Total cases','Cured/Discha
rged/Migrated','Deaths', 'dy_dt', 'd2y_dt2', 'days']]

data.sort_values('Total cases',ascending=False,inplace=True)
sns.set_color_codes("pastel")
sns.barplot(x="Total cases", y="Name of State / UT", data=data,
            label="Total", color="r", ci=None)

sns.set_color_codes("muted")
g =sns.barplot(x="Cured/Discharged/Migrated", y="Name of State / UT", data=data,
            label="Recovered", color="g", ci=None)


# Add a legend and informative axis label
ax.legend(ncol=2, loc="lower right", frameon=True)
ax.set(xlim=(0, 60), ylabel="",
       xlabel="Cases, (dy/dt,d2y/dt2,Days since first case, Total cases)")
sns.despine(left=True, bottom=True)


order = 0
for index, row in data.iterrows():
    val = str(int(row['dy_dt'])) + "," + str(int(row['d2y_dt2'])) + "," + str(in
t(row['days'])) + ","+str(row["Total cases"])
    if val != "0,0,0,0":
        g.text(row["Total cases"], order, val, color='black', ha="center", fonts
ize=9, horizontalalignment='left',  verticalalignment='center')
    order += 1
```



Giving an Index to states based on their handling - where to focus?

Lower the better

In [11]:

```
#Penalty for delay since first case
data["responseIndex"] = (0.001+data["days"])
#Penalty for rate of growth of cases
data["responseIndex"] = data["responseIndex"] + data["responseIndex"] * 1000*(0.001+data["dy_dt"])
#Penalty if the rate of rate itself is up
data["responseIndex"] = data["responseIndex"] + data["responseIndex"] * 1000*(0.001+data["d2y_dt2"])

data["responseIndex"] = data["responseIndex"] + data["responseIndex"] *100*(0.001+data["Total cases"])/(data["Total cases"].sum())

data.sort_values('responseIndex',ascending=False,inplace=True)
data.reset_index()
indexData = data[['Name of State / UT', 'responseIndex']]
indexData.drop(index=0, inplace=True)
display(indexData)
```

	Name of State / UT	responseIndex
6	Rajasthan	2.858837e+06
7	Karnataka	1.713047e+05
5	Tamil Nadu	6.242620e+04
18	Uttarakhand	5.345986e+04
9	Andhra Pradesh	4.013272e+04
10	Odisha	1.820732e+04
8	Gujarat	1.546108e+04
3	West Bengal	1.338055e+04
17	Haryana	5.398962e+02
16	Chhattisgarh	5.639920e+00
29	Puducherry	5.639920e+00
4	Madhya Pradesh	1.053224e-02
19	Himachal Pradesh	7.266939e-03
30	Chandigarh	5.634286e-03
23	Nagaland	4.001633e-03
11	Telangana	4.001633e-03
13	Jharkhand	4.001633e-03
2	Bihar	4.001633e-03
27	Sikkim	4.001633e-03
26	Mizoram	4.001633e-03
25	Arunachal Pradesh	4.001633e-03
24	Goa	4.001633e-03
31	Lakshadweep	4.001633e-03
22	Manipur	4.001633e-03
21	Meghalaya	4.001633e-03
20	Tripura	4.001633e-03
14	Assam	4.001633e-03
28	Delhi	-7.941265e+09
15	Punjab	-9.157845e+09
1	Maharashtra	-2.115659e+10
12	Kerala	-3.711326e+10

Analyze a particular State

In [12]:

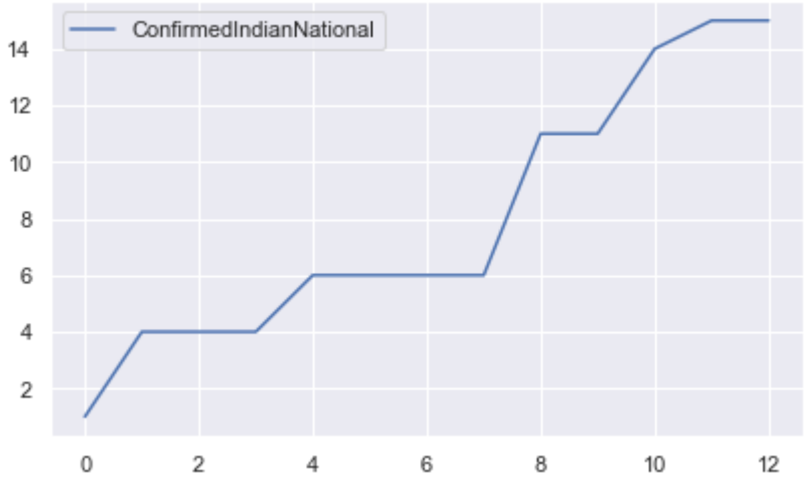
```
covidStateDataDf = covidIndiaDataDf.loc[covidIndiaDataDf["State/UnionTerritory"]
==stateToAnalyze]
```

In [13]:

```
%matplotlib inline
pyplot.figure(1)
pyplot.plot(covidStateDataDf["ConfirmedIndianNational"].values, label="Confirmed
IndianNational")
pyplot.legend()
```

Out[13]:

```
<matplotlib.legend.Legend at 0x1a1d2c0250>
```

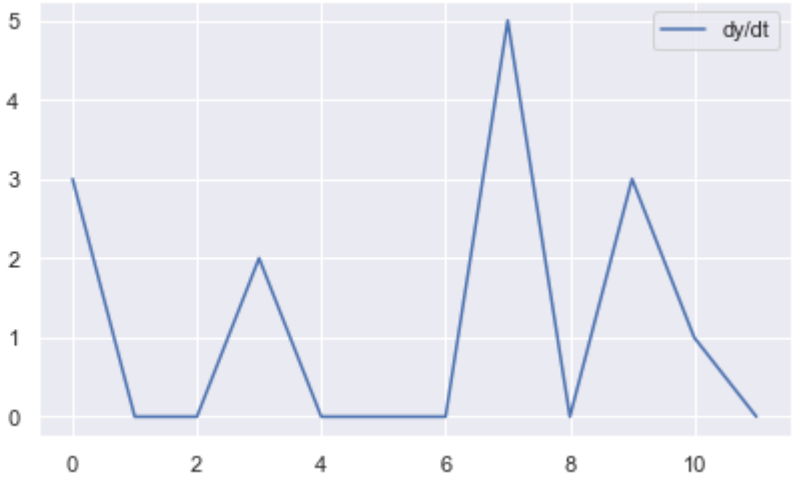


In [14]:

```
dy_dt= np.diff(covidStateDataDf["ConfirmedIndianNational"].values)
pyplot.figure(1)
pyplot.plot(dy_dt, label="dy/dt")
pyplot.legend()
```

Out[14]:

```
<matplotlib.legend.Legend at 0x1a1d7fab50>
```

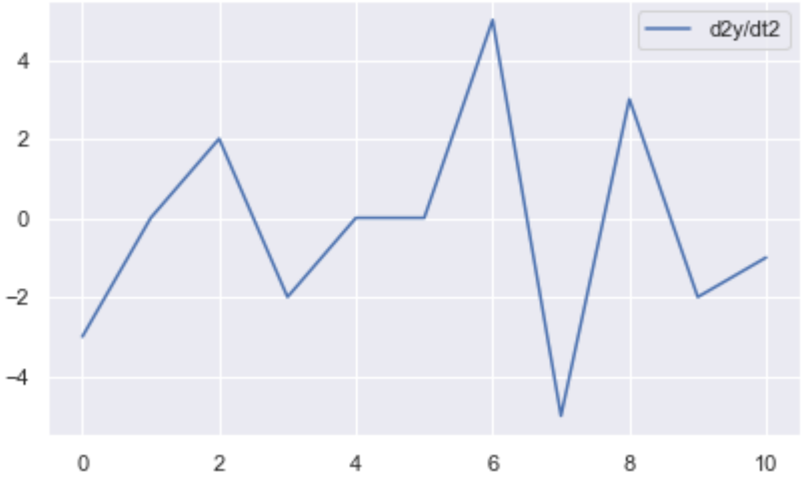


In [15]:

```
d2y_dt2= np.diff(dy_dt)
pyplot.figure(1)
pyplot.plot(d2y_dt2, label="d2y/dt2")
pyplot.legend()
```

Out[15]:

```
<matplotlib.legend.Legend at 0x1a1d7fc710>
```



In [ ]:

```
```

In [ ]:

```
```

SEIR model

It’s an acronym for Susceptible, Exposed, Infected, Recovered

The model classifies the population into four mutually exclusive groups: susceptible (at risk of contracting the disease), exposed (infected but not yet infectious), infectious (capable of transmitting the disease), and removed (those who recover or die from the disease).

$$\dot{S} = -\beta S I \tag{1}$$

$$\dot{E} = \beta S I - \alpha E \tag{2}$$

$$\dot{I} = \alpha E - \gamma I \tag{3}$$

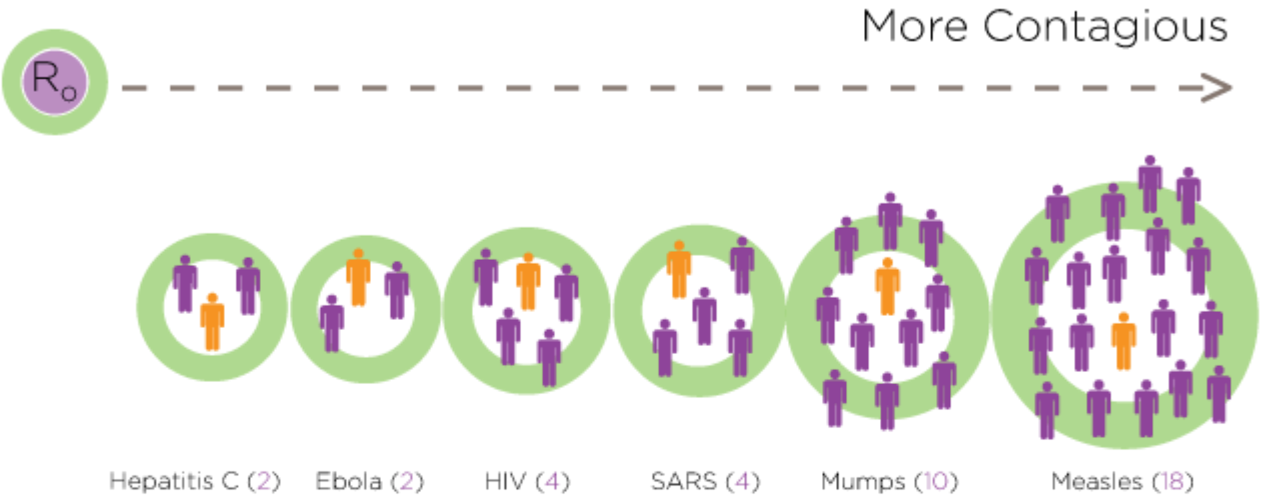
$$\dot{R} = \gamma I \tag{4}$$

$$N = S + E + I + R \tag{5}$$

α is the inverse of the incubation period (1/t_incubation) β is the average contact rate in the population γ is the inverse of the mean infectious period (1/t_infectious)

The final equation, number (5), is a constraint that indicates there are no birth/migration effects in the model; we have a fixed population from beginning to end.

There’s one more parameter we should discuss, the infamous R0 value.



Increasing R0 values indicate more infectious diseases (source: HealthLine.com), This value defines how quickly the disease spreads and can be related to our parameters through the relationship given in Equation (6).

$$R_0 = \frac{\beta}{\gamma} \qquad (6)$$

The differential equations describing SIR model were first derived by Kermack and McKendrick [Proc. R. Soc. A, 115, 772 (1927)]:

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\frac{\beta SI}{N},$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \frac{\beta SI}{N} - \gamma I,$$
$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I.$$

In [1]:

```
import numpy as np
from matplotlib import pyplot
%matplotlib inline

def base_seir_model(init_vals, params, t):
    S_0, E_0, I_0, R_0 = init_vals
    S, E, I, R = [S_0], [E_0], [I_0], [R_0]
    alpha, beta, gamma = params
    dt = t[1] - t[0]
    for _ in t[1:]:
        next_S = S[-1] - (beta*S[-1]*I[-1])*dt
        next_E = E[-1] + (beta*S[-1]*I[-1] - alpha*E[-1])*dt
        next_I = I[-1] + (alpha*E[-1] - gamma*I[-1])*dt
        next_R = R[-1] + (gamma*I[-1])*dt
        S.append(next_S)
        E.append(next_E)
        I.append(next_I)
        R.append(next_R)
    return np.stack([S, E, I, R])
```

Corona virus parameters

A recent study (https://www.thelancet.com/journals/langlo/article/PIIS2214-109X(20)30074-7/fulltext) of COVID-19 estimates some of these values for us (Hellewell et al. 2020), so we can use some of their parameter estimates to get our model off the ground.

Incubation period = 5 days -> α = 0.2

R0 = 3.5

Unfortunately, this paper doesn't provide a value for γ, but we can get an estimate from another paper (https://arxiv.org/pdf/2002.06563.pdf) (which uses a more complex compartmental model) to get our 1/γ value of 2 days, so γ = 0.5.

Plugging the R0 and γ values into Equation (6), we get an estimate of β = 1.75.

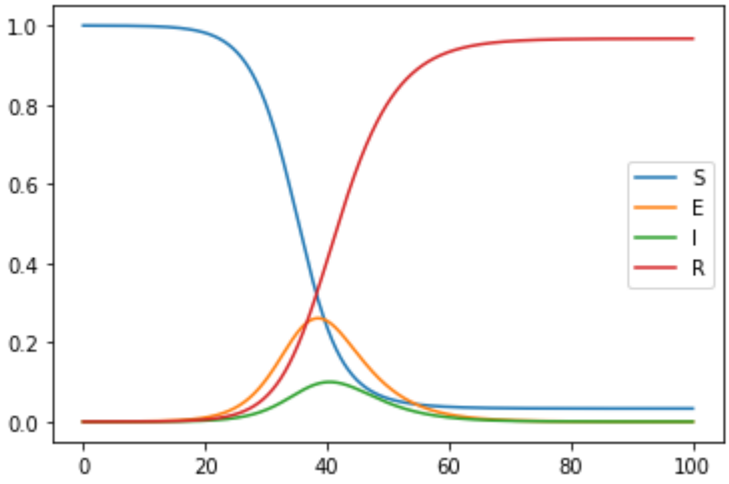
We assume we have 10k people in our population, and we begin with one exposed person and the remaining 9,999 susceptible.

In [2]:

```
# Define parameters
t_max = 100
dt = .1
t = np.linspace(0, t_max, int(t_max/dt) + 1)
N = 10000
init_vals = 1 - 1/N, 1/N, 0, 0
alpha = 0.2
beta = 1.75
gamma = 0.5
params = alpha, beta, gamma
# Run simulation
results = base_seir_model(init_vals, params, t)
```

In [3]:

```
#print(results.shape)
#print(results[:][0].shape)
#print(results[:][1])
#print(results[:][2])
#print(results[:][3])
pyplot.plot(t,results[:][0], label="S" )
pyplot.plot(t,results[:][1], label="E" )
pyplot.plot(t,results[:][2], label="I" )
pyplot.plot(t,results[:][3], label="R" )
pyplot.legend()
pyplot.show()
```



Infected Population

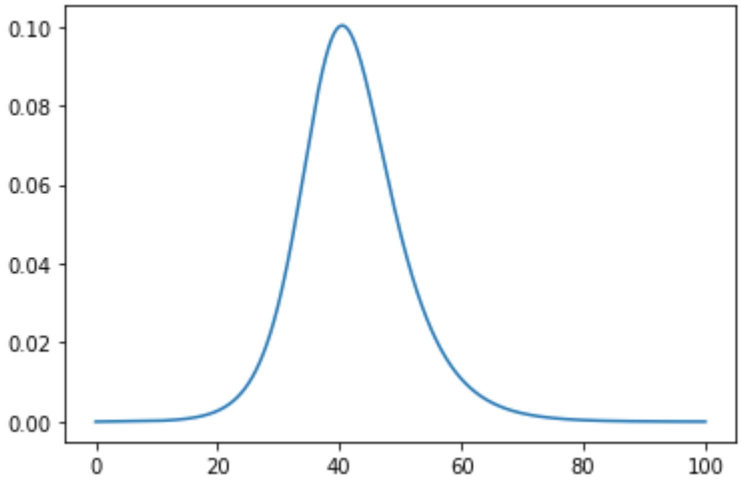
This shows that after 40 days 10% of population will be infected. This could be serious.

In [4]:

```
pyplot.plot(t,results[:][2], label="I" )
```

Out[4]:

```
[<matplotlib.lines.Line2D at 0x11e9ce190>]
```



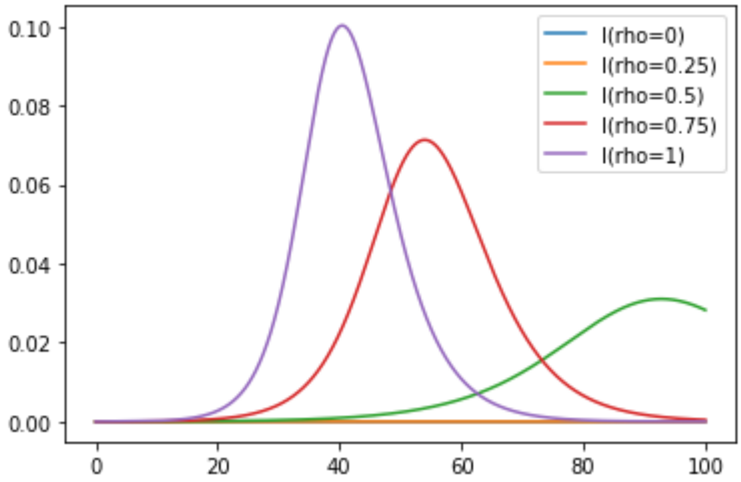
Social Distancing

Social distancing includes avoiding large gatherings, physical contact, and other efforts to mitigate the spread of infectious disease. According to our model, the term this is going to impact is our contact rate, β.

Let’s introduce a new value, ρ, to capture our social distancing effect. This is going to be a constant term between 0–1, where 0 indicates everyone is locked down and quarantined while 1 is equivalent to our base case above.

In [5]:

```
RhoValues = [0, 0.25, 0.5, 0.75, 1]

beta = 1.75

for rho in RhoValues:
    socialBeta = beta*rho
    params = alpha, socialBeta, gamma
    results = base_seir_model(init_vals, params, t)
    pyplot.plot(t,results[:][2], label="I(rho="+str(rho)+")" )

pyplot.legend()
pyplot.show()
```



We go from a base case peak of 10% of the population being infected simultaneously to about 7.5% to a low of 3%. Notice too that it gives people more time to prepare as the peak gets pushed farther out into the future.

These scenarios with social distancing will likely improve the survivability of the disease by giving more time for treatments and supplies to develop while keeping the peaks lower.