In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
import calendar
import plotly.graph_objects as go
import warnings
warnings.filterwarnings("ignore")
import plotly.express as px

%matplotlib inline
```

In [2]:
```
data=pd.read_csv("D:\\cognorise\\unemployment\\Unemployment_Rate_upto_11_2020.csv")
data
```

Out[2]:

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Region.1	longitude	latitude
0	Andhra Pradesh	31-01-2020	M	5.48	16635535	41.02	South	15.9129	79.740
1	Andhra Pradesh	29-02-2020	M	5.83	16545652	40.90	South	15.9129	79.740
2	Andhra Pradesh	31-03-2020	M	5.79	15881197	39.18	South	15.9129	79.740
3	Andhra Pradesh	30-04-2020	M	20.51	11336911	33.10	South	15.9129	79.740
4	Andhra Pradesh	31-05-2020	M	17.43	12988845	36.46	South	15.9129	79.740
...	...	...	...	...	...	...	...	...	...
262	West Bengal	30-06-2020	M	7.29	30726310	40.39	East	22.9868	87.855
263	West Bengal	31-07-2020	M	6.83	35372506	46.17	East	22.9868	87.855
264	West Bengal	31-08-2020	M	14.87	33298644	47.48	East	22.9868	87.855
265	West Bengal	30-09-2020	M	9.35	35707239	47.73	East	22.9868	87.855
266	West Bengal	31-10-2020	M	9.98	33962549	45.63	East	22.9868	87.855

267 rows × 9 columns

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

Out[3]:

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Region.1	longitude	latitude
0	Andhra Pradesh	31-01-2020	M	5.48	16635535	41.02	South	15.9129	79.74
1	Andhra Pradesh	29-02-2020	M	5.83	16545652	40.90	South	15.9129	79.74
2	Andhra Pradesh	31-03-2020	M	5.79	15881197	39.18	South	15.9129	79.74
3	Andhra Pradesh	30-04-2020	M	20.51	11336911	33.10	South	15.9129	79.74
4	Andhra Pradesh	31-05-2020	M	17.43	12988845	36.46	South	15.9129	79.74

In [4]:
```
#updating the column names
data.columns=["State","Date","Frequency","Estimated unemployment rate","Estimated employed",
"Estimated labour participation rate","Region","Longitude","Latitude"]
```

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

Out[5]:

	State	Date	Frequency	Estimated unemployment rate	Estimated employed	Estimated labour participation rate	Region	Longitude	Latitude
0	Andhra Pradesh	31-01-2020	M	5.48	16635535	41.02	South	15.9129	79.74
1	Andhra Pradesh	29-02-2020	M	5.83	16545652	40.90	South	15.9129	79.74
2	Andhra Pradesh	31-03-2020	M	5.79	15881197	39.18	South	15.9129	79.74
3	Andhra Pradesh	30-04-2020	M	20.51	11336911	33.10	South	15.9129	79.74
4	Andhra Pradesh	31-05-2020	M	17.43	12988845	36.46	South	15.9129	79.74

In [6]:
```
data.shape
```

Out[6]:
```
(267, 9)
```

In [7]:
```
data.columns
```

Out[7]:
```
Index(['State', 'Date', 'Frequency', 'Estimated unemployment rate',
       'Estimated employed', 'Estimated labour participation rate', 'Region',
       'Longitude', 'Latitude'],
      dtype='object')
```

In [8]:
```
data.describe()
```

Out[8]:

	Estimated unemployment rate	Estimated employed	Estimated labour participation rate	Longitude	Latitude
count	267.000000	2.670000e+02	267.000000	267.000000	267.000000
mean	12.236929	1.396211e+07	41.681573	22.826048	80.532425
std	10.803283	1.336632e+07	7.845419	6.270731	5.831738
min	0.500000	1.175420e+05	16.770000	10.850500	71.192400
25%	4.845000	2.838930e+06	37.265000	18.112400	76.085600
50%	9.650000	9.732417e+06	40.390000	23.610200	79.019300
75%	16.755000	2.187869e+07	44.055000	27.278400	85.279900
max	75.850000	5.943376e+07	69.690000	33.778200	92.937600

In [9]:
```
data.dtypes
```

Out[9]:
```
State                                   object
Date                                    object
Frequency                               object
Estimated unemployment rate            float64
Estimated employed                       int64
Estimated labour participation rate    float64
Region                                  object
Longitude                              float64
Latitude                               float64
dtype: object
```

In [10]:

```
data['Date']=pd.to_datetime(data["Date"])
```

In [11]:

```
data.dtypes
```

Out[11]:

```
State                                          object
Date                                   datetime64[ns]
Frequency                                      object
Estimated unemployment rate                   float64
Estimated employed                              int64
Estimated labour participation rate           float64
Region                                         object
Longitude                                     float64
Latitude                                      float64
dtype: object
```

In [12]:

```
data.isnull().sum()
```

Out[12]:

```
State                                  0
Date                                   0
Frequency                              0
Estimated unemployment rate            0
Estimated employed                     0
Estimated labour participation rate    0
Region                                 0
Longitude                              0
Latitude                               0
dtype: int64
```

In [13]:

```
data.duplicated().any()
```

Out[13]:

```
False
```

In [14]:

```
#Converting 'Frequency' and 'Region' columns to categorical data type
data['Frequency'] = data['Frequency'].astype('category')
data['Region'] = data['Region'].astype('category')
```

In [15]:
```
data.dtypes
```

Out[15]:
```
State                                          object
Date                                   datetime64[ns]
Frequency                                    category
Estimated unemployment rate                   float64
Estimated employed                              int64
Estimated labour participation rate           float64
Region                                       category
Longitude                                     float64
Latitude                                      float64
dtype: object
```

In [16]:
```
#extract month
data["month"]=data["Date"].dt.month
```

In [17]:
```
#converting 'month' to integer format
data['Month_int'] = data['month'].apply(lambda x: int(x))
# Mapping integer month values to abbreviated month names
data['Month_name'] = data['Month_int'].apply(lambda x: calendar.month_abbr[x])
data['Month'] = data['Month_int'].apply(lambda x: calendar.month_abbr[x])
```

In [18]:

```
data.tail()
```

Out[18]:

	State	Date	Frequency	Estimated unemployment rate	Estimated employed	Estimated labour participation rate	Region	Longitude	Latitude	month	Month_int	Month_name
262	West Bengal	2020-06-30	M	7.29	30726310	40.39	East	22.9868	87.855	6	6	Jun
263	West Bengal	2020-07-31	M	6.83	35372506	46.17	East	22.9868	87.855	7	7	Jul
264	West Bengal	2020-08-31	M	14.87	33298644	47.48	East	22.9868	87.855	8	8	Aug
265	West Bengal	2020-09-30	M	9.35	35707239	47.73	East	22.9868	87.855	9	9	Sep
266	West Bengal	2020-10-31	M	9.98	33962549	45.63	East	22.9868	87.855	10	10	Oct

In [19]:

```
#Basic Statistics
data_stats = data[['Estimated unemployment rate', 'Estimated employed', 'Estimated labour participation rate'
round(data_stats.describe().T, 2)
```

Out[19]:

	count	mean	std	min	25%	50%	75%	max
Estimated unemployment rate	267.0	12.24	10.80	0.50	4.84	9.65	16.76	75.85
Estimated employed	267.0	13962105.72	13366318.36	117542.00	2838930.50	9732417.00	21878686.00	59433759.00
Estimated labour participation rate	267.0	41.68	7.85	16.77	37.26	40.39	44.06	69.69

In [20]:

```
region_stats = data.groupby(['Region'])[['Estimated unemployment rate', 'Estimated employed',
'Estimated labour participation rate']].mean().reset_index()
round(region_stats, 2)
```

Out[20]:

	Region	Estimated unemployment rate	Estimated employed	Estimated labour participation rate
0	East	13.92	19602366.90	40.11
1	North	15.89	13072487.92	38.70
2	Northeast	10.95	3617105.53	52.06
3	South	10.45	14040589.33	40.44
4	West	8.24	18623512.72	41.26

In [21]:

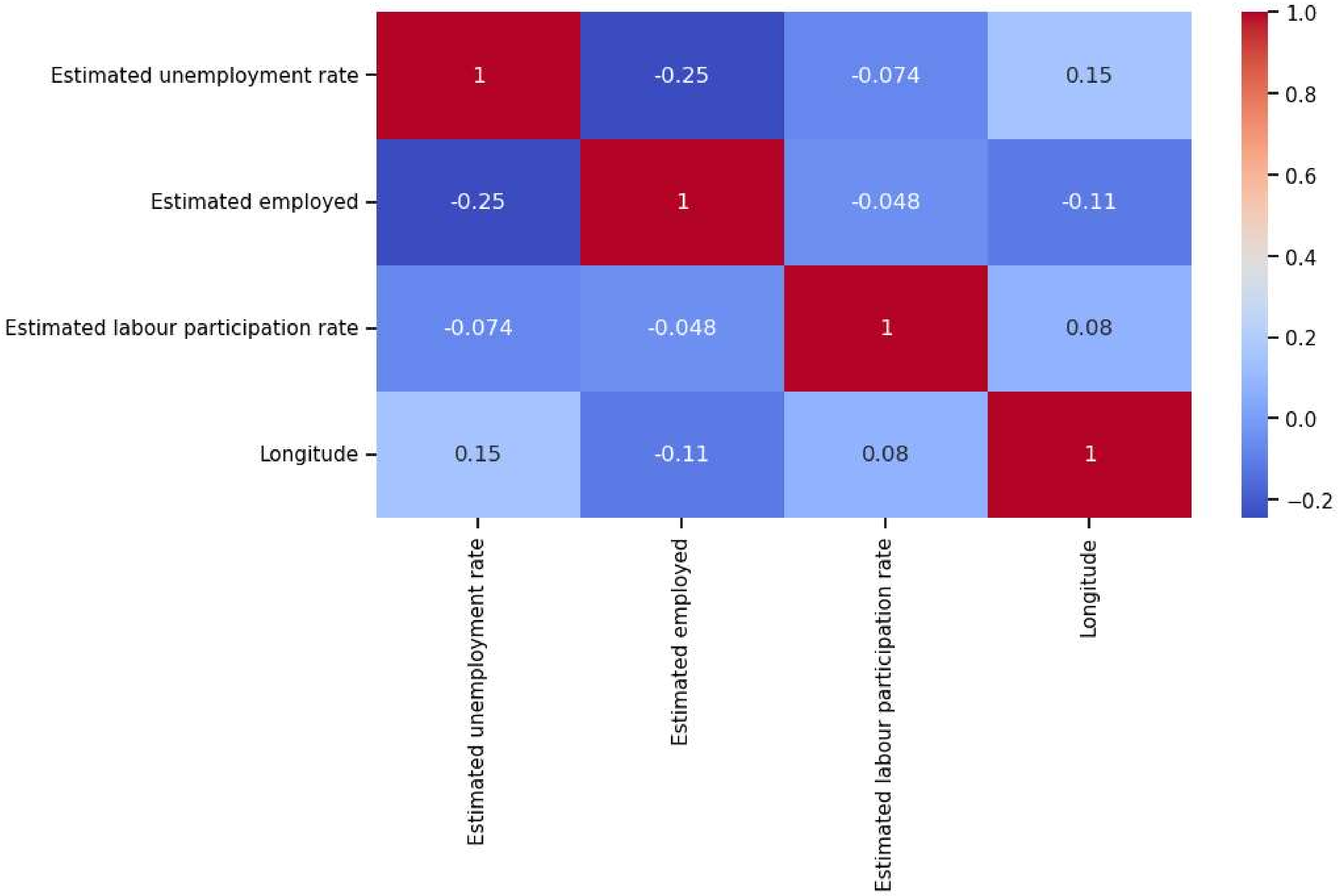
```
IMD = data.groupby(["Month"])[['Estimated unemployment rate','Estimated employed','Estimated labour participa
IMD = pd.DataFrame(IMD).reset_index()
```

In [22]:

```
State = data.groupby("State")[['Estimated unemployment rate','Estimated employed','Estimated labour participa
State = pd.DataFrame(State).reset_index()
```

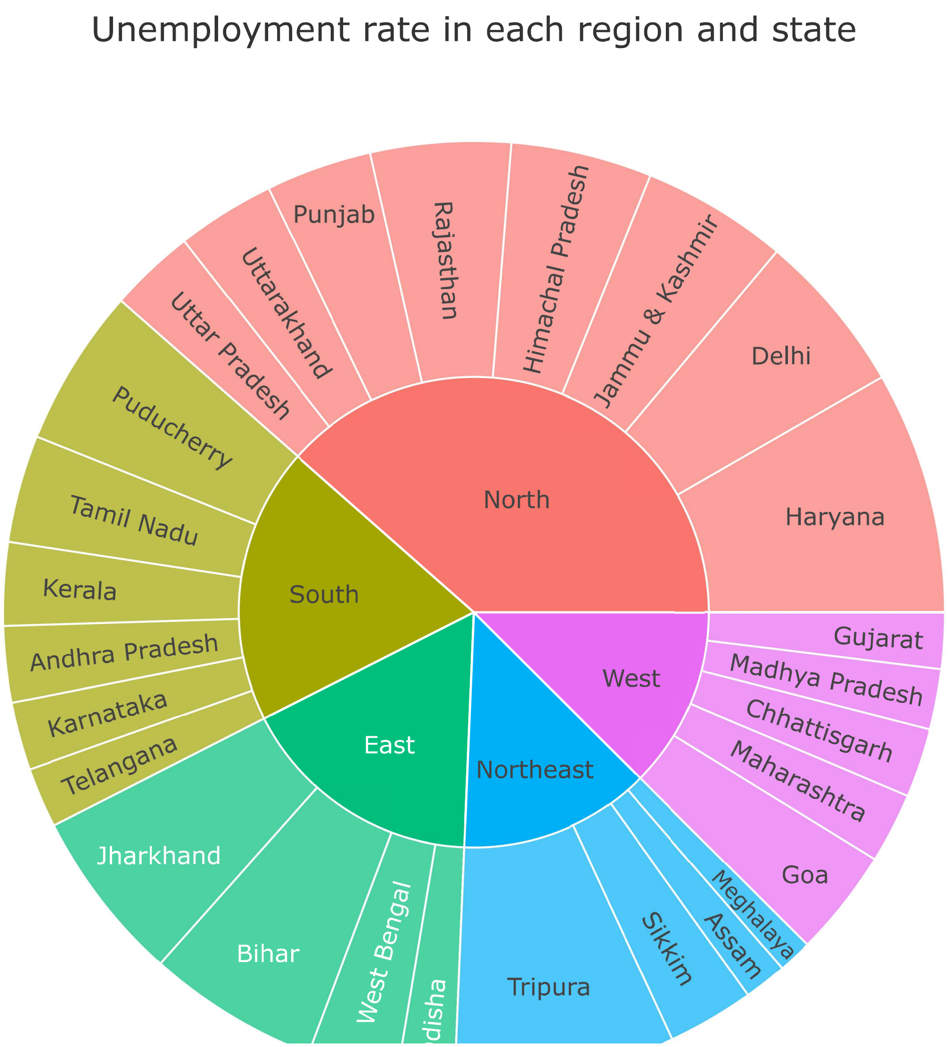
In [23]:
```
heat_maps = data[["Estimated unemployment rate", "Estimated employed","Estimated labour participation rate",'
heat_maps = heat_maps.corr()
plt.figure(figsize=(10,5))
sns.set_context("notebook",font_scale=1)
sns.heatmap(heat_maps,annot=True , cmap='coolwarm')
```

Out[23]:
```
<Axes: >
```



In [24]:

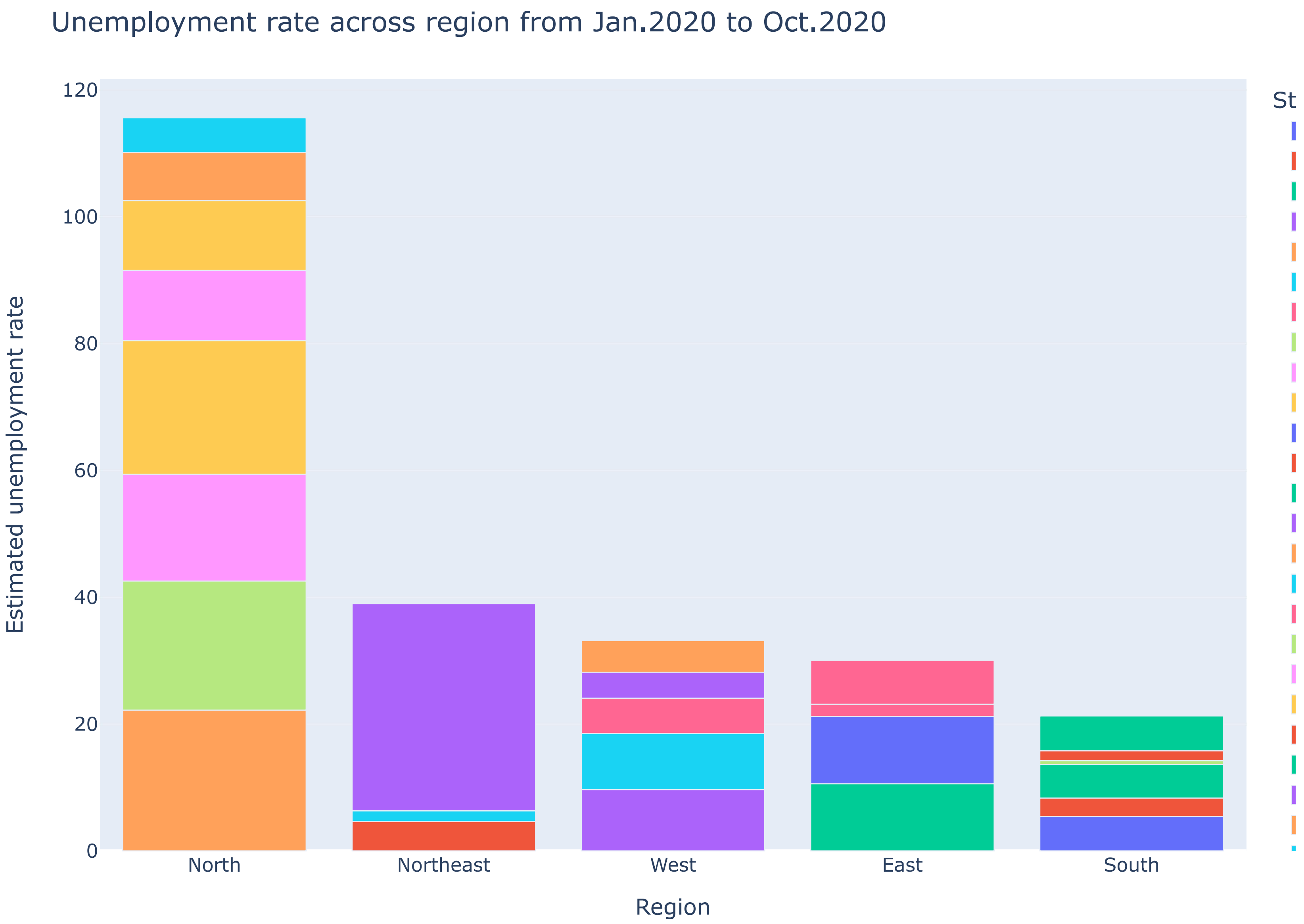
```
# Sunburst chart showing unemployment rate in each region and state
unemplo_data = data[['State', 'Region', 'Estimated unemployment rate', 'Estimated employed', 'Estimated labou
unemplo = unemplo_data.groupby(['Region', 'State'])['Estimated unemployment rate'].mean().reset_index()
fig = px.sunburst(unemplo, path=['Region', 'State'], values='Estimated unemployment rate',
color_continuous_scale='Plasma', title='Unemployment rate in each region and state',
height=650, template='ggplot2')
fig.show()
```


Unemployment rate in each region and state
North
Haryana
Delhi
Jammu & Kashmir
Himachal Pradesh
Rajasthan
Punjab
Uttarakhand
Uttar Pradesh
South
Puducherry
Tamil Nadu
Kerala
Andhra Pradesh
Karnataka
Telangana
East
Jharkhand
Bihar
West Bengal
disha
Northeast
Tripura
Sikkim
Assam
Meghalaya
West
Gujarat
Madhya Pradesh
Chhattisgarh
Maharashtra
Goa

In [25]:

```
fig = px.bar(data, x='Region', y='Estimated unemployment rate', animation_frame='Month_name', color='State')
fig.update_layout(title='Unemployment rate across region from Jan.2020 to Oct.2020', height=700, template='pl

fig.update_layout(xaxis={'categoryorder': 'total descending'})
fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 2000
fig.show()
fig = px.bar(data, x='Region', y='Estimated unemployment rate', animation_frame='Month_name', color='State')
```


Unemployment rate across region from Jan.2020 to Oct.2020
Estimated unemployment rate
0
20
40
60
80
100
120
North
Northeast
West
East
South
Region
St

In [26]:

```
#data representation before and after the lockdown
before_lockdown = data[(data['Month_int']>=1) & (data['Month_int']<4)]
after_lockdown = data[(data['Month_int']>=4) & (data['Month_int']<=6)]
```

In [27]:

```
data.Region.unique()
```

Out[27]:

```
['South', 'Northeast', 'East', 'West', 'North']
Categories (5, object): ['East', 'North', 'Northeast', 'South', 'West']
```

In [28]:

```
af_lockdown=after_lockdown.groupby('State')['Estimated unemployment rate'].mean().reset_index()
lockdown= before_lockdown.groupby('State')['Estimated unemployment rate'].mean().reset_index()
lockdown['Unemployment Rate before lockdown'] = af_lockdown['Estimated unemployment rate']
lockdown.columns=['State','Unemployment Rate Before Lockdown','Unemployment Rate After Lockdown']
lockdown.head()
```

Out[28]:

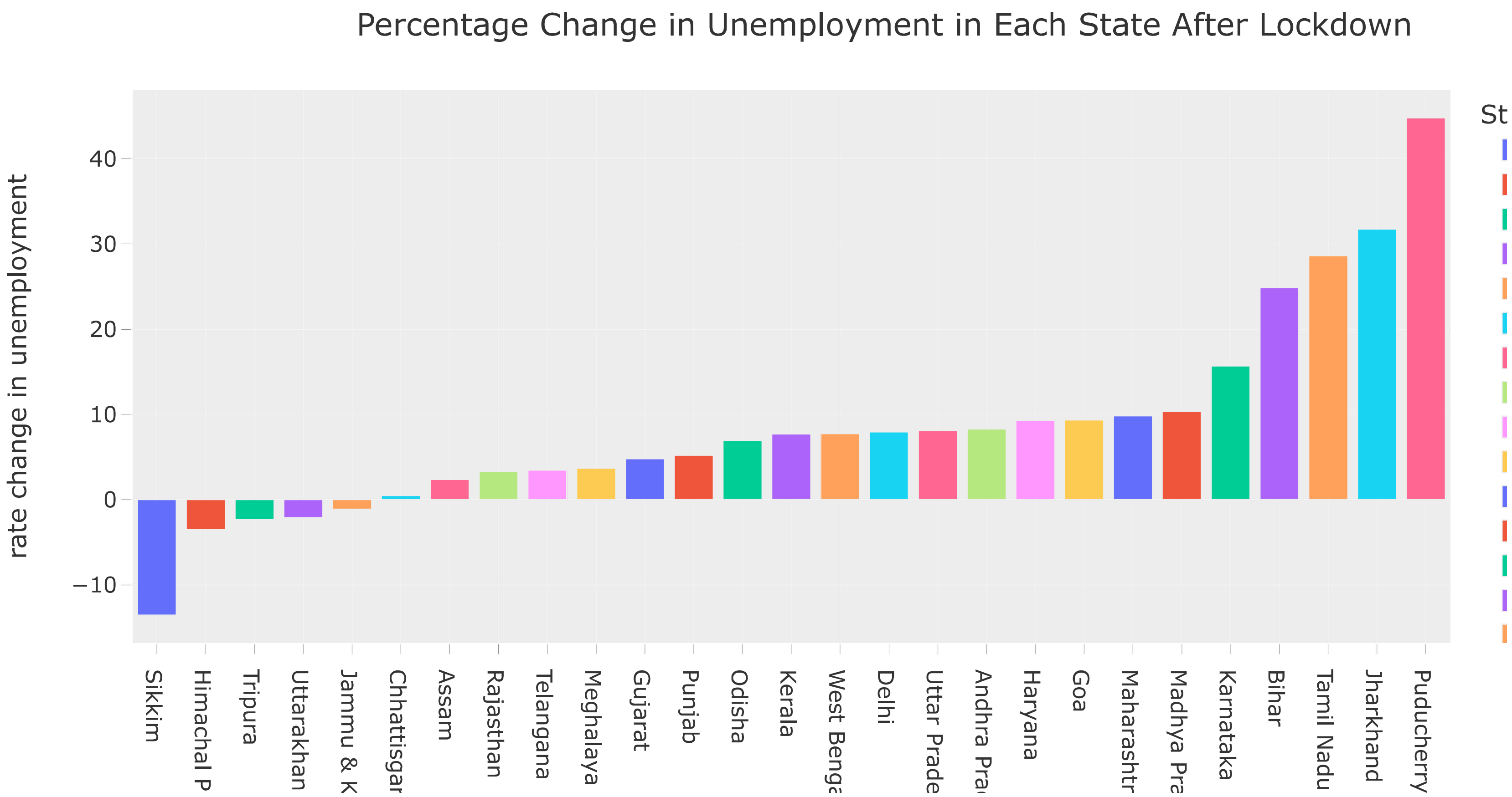
	State	Unemployment Rate Before Lockdown	Unemployment Rate After Lockdown
0	Andhra Pradesh	5.700000	13.750000
1	Assam	4.613333	7.070000
2	Bihar	12.110000	36.806667
3	Chhattisgarh	8.523333	9.380000
4	Delhi	18.036667	25.713333

In [29]:

```
# percentage change in unemployment rate
lockdown['rate change in unemployment'] = round(lockdown['Unemployment Rate After Lockdown']) -lockdown['Unem
plot_per = lockdown.sort_values('rate change in unemployment')
```

In [30]:

```
# percentage change in unemployment after lockdown
fig = px.bar(plot_per, x='State',y='rate change in unemployment',color='State')
fig.update_layout(
    title='Percentage Change in Unemployment in Each State After Lockdown',
    template='ggplot2'
)
fig.show()
```



In [31]:

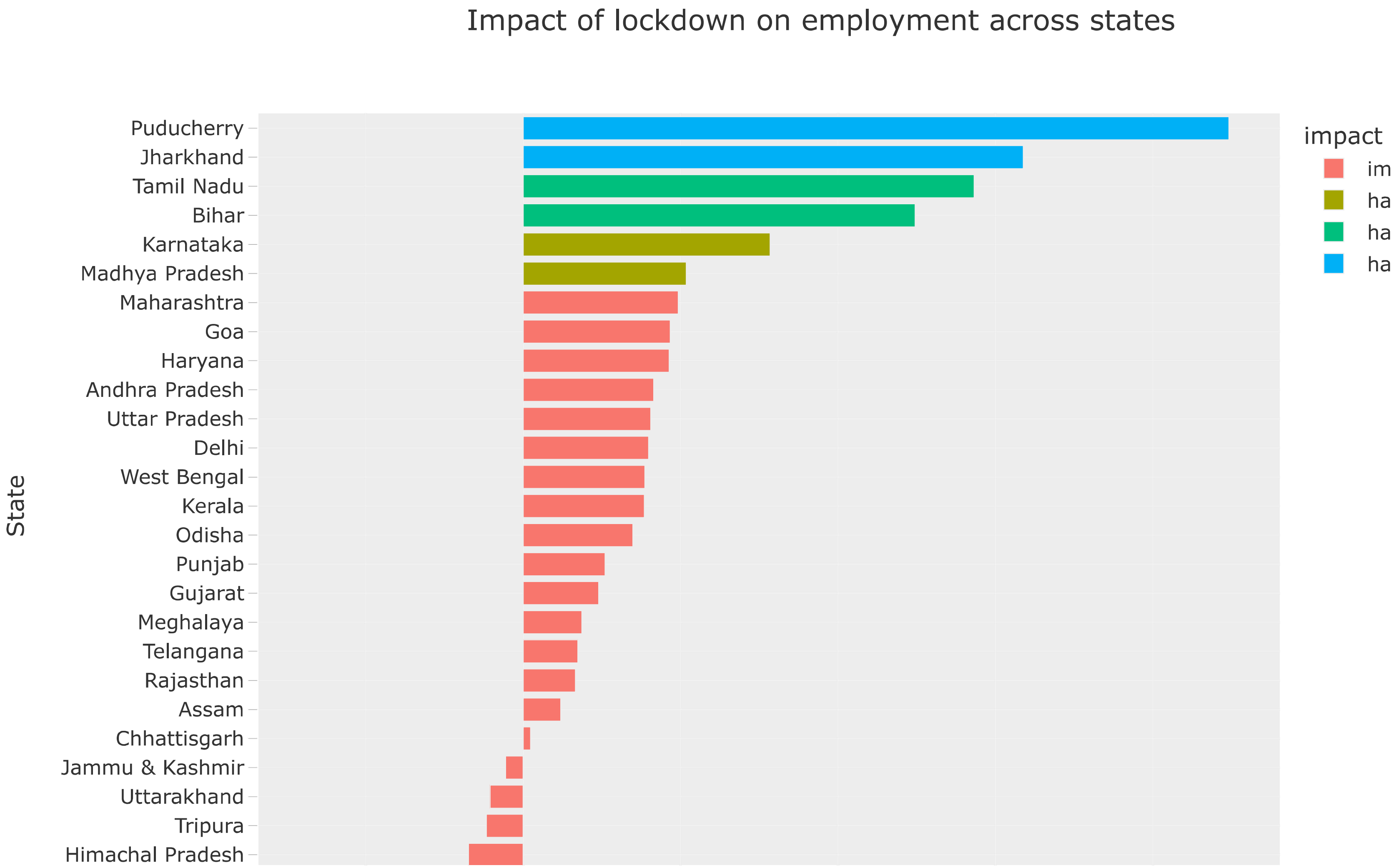
```
# function to sort value based on impact
def sort_impact(x):
   if x <= 10:
      return 'impacted States'
   elif x <= 20:
      return 'hard impacted States'
   elif x <= 30:
      return 'harder impacted States'
   elif x <= 46:
      return 'hardest impacted States'
   return x
```

In [32]:

```
plot_per['impact status'] = plot_per['rate change in unemployment'].apply(lambda x:sort_impact(x))
```

In [33]:

```
fig = px.bar(plot_per, y='State',x='rate change in unemployment',color='impact status',
title='Impact of lockdown on employment across states',template='ggplot2',height=650)
fig.show()
```


Impact of lockdown on employment across states
State
Puducherry
Jharkhand
Tamil Nadu
Bihar
Karnataka
Madhya Pradesh
Maharashtra
Goa
Haryana
Andhra Pradesh
Uttar Pradesh
Delhi
West Bengal
Kerala
Odisha
Punjab
Gujarat
Meghalaya
Telangana
Rajasthan
Assam
Chhattisgarh
Jammu & Kashmir
Uttarakhand
Tripura
Himachal Pradesh
impact
im
ha
ha
ha

In [34]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,6))
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```

In [ ]: